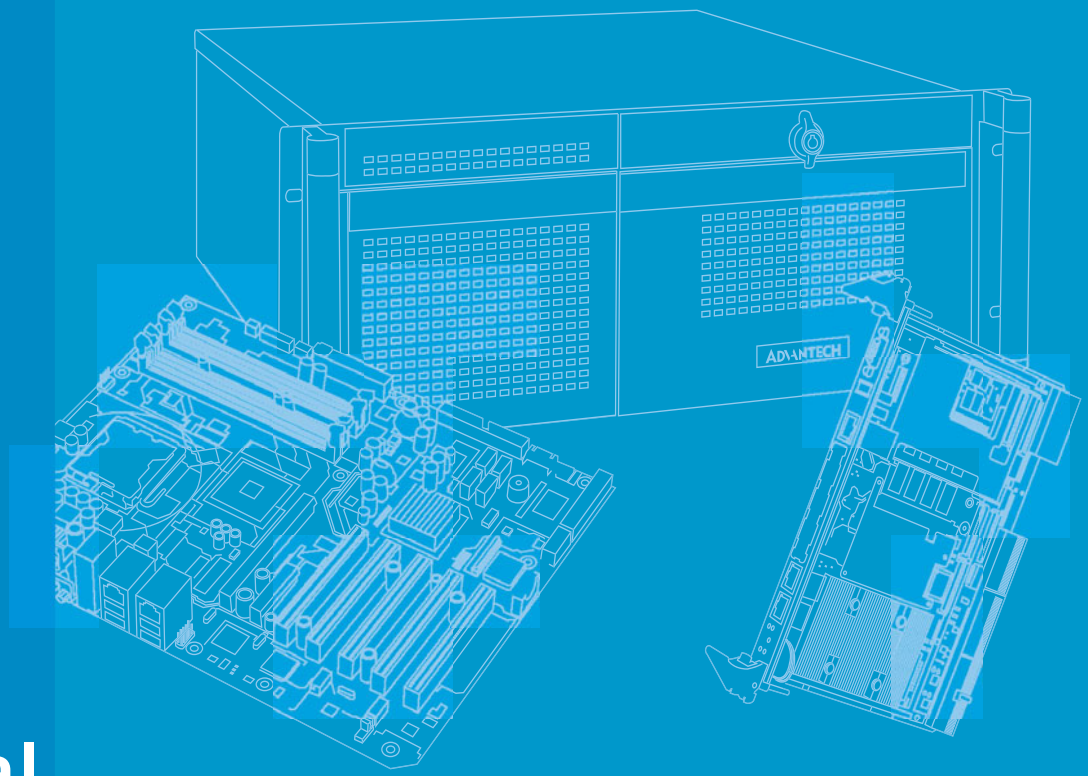

User Manual

DVP- SDK

Capture SDK
Encoeder SDK
Playback SDK

ADVANTECH

Enabling an Intelligent Planet

Copyright

The documentation and the software included with this product are copyrighted 2012 by Advantech Co., Ltd. All rights are reserved. Advantech Co., Ltd. reserves the right to make improvements in the products described in this manual at any time without notice. No part of this manual may be reproduced, copied, translated or transmitted in any form or by any means without the prior written permission of Advantech Co., Ltd. Information provided in this manual is intended to be accurate and reliable. However, Advantech Co., Ltd. assumes no responsibility for its use, nor for any infringements of the rights of third parties, which may result from its use.

Acknowledgements

Intel and Pentium are trademarks of Intel Corporation.

Microsoft Windows and MS-DOS are registered trademarks of Microsoft Corp.

All other product names or trademarks are properties of their respective owners.

Product Warranty (2 years)

Advantech warrants to you, the original purchaser, that each of its products will be free from defects in materials and workmanship for two years from the date of purchase.

This warranty does not apply to any products which have been repaired or altered by persons other than repair personnel authorized by Advantech, or which have been subject to misuse, abuse, accident or improper installation. Advantech assumes no liability under the terms of this warranty as a consequence of such events.

Because of Advantech's high quality-control standards and rigorous testing, most of our customers never need to use our repair service. If an Advantech product is defective, it will be repaired or replaced at no charge during the warranty period. For out-of-warranty repairs, you will be billed according to the cost of replacement materials, service time and freight. Please consult your dealer for more details.

If you think you have a defective product, follow these steps:

1. Collect all the information about the problem encountered. (For example, CPU speed, Advantech products used, other hardware and software used, etc.) Note anything abnormal and list any onscreen messages you get when the problem occurs.
2. Call your dealer and describe the problem. Please have your manual, product, and any helpful information readily available.
3. If your product is diagnosed as defective, obtain an RMA (return merchandize authorization) number from your dealer. This allows us to process your return more quickly.
4. Carefully pack the defective product, a fully-completed Repair and Replacement Order Card and a photocopy proof of purchase date (such as your sales receipt) in a shippable container. A product returned without proof of the purchase date is not eligible for warranty service.
5. Write the RMA number visibly on the outside of the package and ship it prepaid to your dealer.

Part No. XXXXXXXXXX Edition 1

Printed in Taiwan September 2012

Declaration of Conformity

CE

This product has passed the CE test for environmental specifications when shielded cables are used for external wiring. We recommend the use of shielded cables. This kind of cable is available from Advantech. Please contact your local supplier for ordering information.

CE

This product has passed the CE test for environmental specifications. Test conditions for passing included the equipment being operated within an industrial enclosure. In order to protect the product from being damaged by ESD (Electrostatic Discharge) and EMI leakage, we strongly recommend the use of CE-compliant industrial enclosure products.

FCC Class A

Note: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

FCC Class B

Note: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

FM

This equipment has passed the FM certification. According to the National Fire Protection Association, work sites are classified into different classes, divisions and groups, based on hazard considerations. This equipment is compliant with the specifications of Class I, Division 2, Groups A, B, C and D indoor hazards.

Technical Support and Assistance

1. Visit the Advantech website at http://support.advantech.com where you can find the latest information about the product.
2. Contact your distributor, sales representative, or Advantech's customer service center for technical support if you need additional assistance. Please have the following information ready before you call:
   - Product name and serial number
   - Description of your peripheral attachments
   - Description of your software (operating system, version, application software, etc.)
   - A complete description of the problem
   - The exact wording of any error messages

Warnings, Cautions and Notes

Warning! *Warnings indicate conditions, which if not observed, can cause personal injury!*

Caution! *Cautions are included to help you avoid damaging hardware or losing data. e.g.*

There is a danger of a new battery exploding if it is incorrectly installed. Do not attempt to recharge, force open, or heat the battery. Replace the battery only with the same or equivalent type recommended by the manufacturer. Discard used batteries according to the manufacturer's instructions.

Note! *Notes provide optional additional information.*

Document Feedback

To assist us in making improvements to this manual, we would welcome comments and constructive criticism. Please send all such - in writing to: support@advantech.com

Packing List

Before setting up the system, check that the items listed below are included and in good condition. If any item does not accord with the table, please contact your dealer immediately.

- Item XXXXXXXX
- Box XXXXXXXXX

Safety Instructions

1. Read these safety instructions carefully.
2. Keep this User Manual for later reference.
3. Disconnect this equipment from any AC outlet before cleaning. Use a damp cloth. Do not use liquid or spray detergents for cleaning.
4. For plug-in equipment, the power outlet socket must be located near the equipment and must be easily accessible.
5. Keep this equipment away from humidity.
6. Put this equipment on a reliable surface during installation. Dropping it or letting it fall may cause damage.
7. The openings on the enclosure are for air convection. Protect the equipment from overheating. DO NOT COVER THE OPENINGS.
8. Make sure the voltage of the power source is correct before connecting the equipment to the power outlet.
9. Position the power cord so that people cannot step on it. Do not place anything over the power cord.
10. All cautions and warnings on the equipment should be noted.
11. If the equipment is not used for a long time, disconnect it from the power source to avoid damage by transient overvoltage.
12. Never pour any liquid into an opening. This may cause fire or electrical shock.
13. Never open the equipment. For safety reasons, the equipment should be opened only by qualified service personnel.
14. If one of the following situations arises, get the equipment checked by service personnel:
    - The power cord or plug is damaged.
    - Liquid has penetrated into the equipment.
    - The equipment has been exposed to moisture.
    - The equipment does not work well, or you cannot get it to work according to the user's manual.
    - The equipment has been dropped and damaged.
    - The equipment has obvious signs of breakage.
15. DO NOT LEAVE THIS EQUIPMENT IN AN ENVIRONMENT WHERE THE STORAGE TEMPERATURE MAY GO BELOW -20° C (-4° F) OR ABOVE 60° C (140° F). THIS COULD DAMAGE THE EQUIPMENT. THE EQUIPMENT SHOULD BE IN A CONTROLLED ENVIRONMENT.
16. CAUTION: DANGER OF EXPLOSION IF BATTERY IS INCORRECTLY REPLACED. REPLACE ONLY WITH THE SAME OR EQUIVALENT TYPE RECOMMENDED BY THE MANUFACTURER, DISCARD USED BATTERIES ACCORDING TO THE MANUFACTURER'S INSTRUCTIONS.

The sound pressure level at the operator's position according to IEC 704-1:1982 is no more than 70 dB (A).

DISCLAIMER: This set of instructions is given according to IEC 704-1. Advantech disclaims all responsibility for the accuracy of any statements contained herein.

Safety Precaution - Static Electricity

Follow these simple precautions to protect yourself from harm and the products from damage.

- To avoid electrical shock, always disconnect the power from your PC chassis before you work on it. Don't touch any components on the CPU card or other cards while the PC is on.
- Disconnect power before making any configuration changes. The sudden rush of power as you connect a jumper or install a card may damage sensitive electronic components.

Contents

Chapter 1 The SDK Packages 1

1.1 System Requirements 2

Chapter 2 Overview 3

2.1 Capture APIs 4
2.2 Encoding APIs 6
2.3 Playback APIs 8

Chapter 3 Functions Description 9

3.1 Capture APIs 10
3.1.1 Data Type 10
3.1.2 Methods 10
3.2 Encoding APIs 59
3.2.1 Data Type 59
3.2.2 Callback 60
3.2.3 Structures 61
3.2.4 Methods 63
3.3 Playback APIs 88
3.3.1 Data Type 88
3.3.2 Methods 89

Chapter 1

The SDK Packages

1.1 System Requirements

- **Platform:**
  Windows XP, Windows Vista, Windows 7 32 bits
- **Application:**
  The following designing environments suit for a software programmer, for example, Microsoft VC 6.0, VC++ 2003 / 2005

Chapter 2

Overview

2.1 Capture APIs

AdvDVP- Series Functions Library
Library: DVP7010B.DLL

Data Type Summary

Res	The function returns value

Method Summary

Initialize and close	
AdvDVP_CreateSDKInstance	Creates a SDK instance of capture object
AdvDVP_InitSDK	Initializes and starts the SDK
AdvDVP_CloseSDK	Closes up the SDK

Capture control	
AdvDVP_GetNoOfDevices	Get the number of video capture devices
AdvDVP_Start	Starts video capturing
AdvDVP_Stop	Stops video capturing
AdvDVP_GetCapState	Gets video capture state
AdvDVP_SetNewFrameCallback	Sets a call back function for video RAW data
AdvDVP_GetCurFrameBuffer	Gets RAW data of current video frame
AdvDVP_GetCurFrameSize	Gets the width and height of current video frame
AdvDVP_GetNoOfAudios	Gets the number of audio capture devices
AdvDVP_StartAudio	Starts audio capturing
AdvDVP_StopAudio	Stops audio capturing
AdvDVP_GetCapAudioState	Gets audio capture state
AdvDVP_SetAudioCallback	Sets a call back function for audio RAW data
AdvDVP_GetNoOfChannels	Gets the number of video channels
AdvDVP_GetCurChannel	Gets the current channel number of video device
AdvDVP_ChannelToDevice	Retrieves the video device number from video channel number

Capture setting	
AdvDVP_GetVideoFormat	Gets video input format
AdvDVP_SetVideoFormat	Sets video input format
AdvDVP_GetFrameRate	Gets video frame rate (fps)
AdvDVP_SetFrameRate	Sets video frame rate
AdvDVP_GetResolution	Gets video resolution
AdvDVP_SetResolution	Sets video resolution
AdvDVP_GetVideoInput	Gets video input MUX
AdvDVP_SetVideoInput	Sets video input MUX
AdvDVP_GetVideoSwitching	Gets the number of video MUX and switching channels
AdvDVP_GetMappingResolution	Retrieves the video and height from video size ID
AdvDVP_GetAudioFormat	Gets the format of audio
AdvDVP_SetAudioFormat	Sets the format of audio

AdvDVP_GetAudioVolume	Gets the volume of audio
AdvDVP_SetAudioVolume	Sets the volume of audio
AdvDVP_SetAVSyncChannel	Sets the audio channel synchronize with video channel

Sensor control	
AdvDVP_GetBrightness	Gets the value of brightness
AdvDVP_SetBrightness	Sets the value of brightness
AdvDVP_GetContrast	Gets the value of contrast
AdvDVP_SetContrast	Sets the value of contrast
AdvDVP_GetHue	Get the value of hue
AdvDVP_SetHue	Set the value of hue
AdvDVP_GetSaturation	Get the value of saturation
AdvDVP_SetSaturation	Set the value of saturation

GPIO control	
AdvDVP_GPIOGetData	Gets the state of GPIO pin
AdvDVP_GPIOSetData	Sets the state of GPIO pin

De-interlace control	
AdvDVP_GetDeinterlace	Gets the enabling flag of de-interlace function
AdvDVP_SetDeinterlace	Enables video de-interlace function or not

Text OSD	
AdvDVP_GetNumOfOSD	Gets the supported video OSD number of video channel
AdvDVP_SetTextOSDEx	Sets / Enables the video OSD function
AdvDVP_ClearTextOSDEx	Clears / Disables the video OSD function

Compression control	
AdvDVP_StartCompress	Starts H/W compression of video
AdvDVP_StopCompress	Stops H/W compression of video
AdvDVP_GetCompressState	Gets video H/W compression state
AdvDVP_RestartCompress	Restart / separate video H/W compression stream
AdvDVP_SetCompressCallback	Sets a callback function of H/W compression stream

Compression control	
AdvDVP_GetVideoCompressFormats	Gets the video H/W compression formats list
AdvDVP_GetCurVideoCompressFormat	Gets current video H/W compression format
AdvDVP_SetVideoCompressFormat	Sets the video H/W compression format
AdvDVP_IsSupportCompress	Checks if supported video H/W compression format or not
AdvDVP_GetCompressAttr	Gets the attributes of H/W compression
AdvDVP_SetCompressAttr	Sets the attributes of H/W compression

Miscellaneous	
AdvDVP_IsVideoPresent	Checks if the video preset or not
AdvDVP_SetVideoZoom	Sets zoom in/out of video preview
AdvDVP_EnableWatchDog	Enables the watchdog detection
AdvDVP_ResetWatchDog	Reset the watchdog counting
AdvDVP_IsSupportVideoFormat	Checks if supported video input format or not

“C” exports	
AdvDVP_CreateSDKInstance_C	Creates and gets a SDK instance of capture object
AdvDVP_*XXXX*_C	*XXXX* means all of capture function names

2.2 Encoding APIs

AdvDVP- Series Encoding Functions Library

Library: DVP7010BEnc.DLL

Data Type Summary

EncRes	The function returns value
PSTREAMREADBEGIN	The “stream read” begin function pointer
PSTREAMREADPROC	The “stream read” process function pointer
PSTREAMREADEND	The “stream read” end function pointer
STREAMREAD STRUCT	The structure stores the “stream read” callback function pointers

Method Summary

Initialize and close	
AdvDVP_CreateEncSDKInstance	Creates a SDK instance of encoder object
AdvDVP_InitSDK	Initializes and starts the SDK
AdvDVP_CloseSDK	Closes up the SDK
AdvDVP_InitEncoder	Opens and initializes A/V encoder
AdvDVP_CloseEncoder	Closes and releases A/V encoder

Encode control	
AdvDVP_StartVideoEncoder	Starts video encoding
AdvDVP_VideoEncode	Encodes one video frame
AdvDVP_StopVideoEncode	Stops video encoding
AdvDVP_GetState	Gets encoder state
AdvDVP_SetStreamReadCB	Sets the “stream read” callback functions
AdvDVP_StartAudioEncoder	Starts audio encoding
AdvDVP_AudioEncode	Encodes one audio frame
AdvDVP_StopAudioEncode	Stops audio encoding

Encode setting	
AdvDVP_GetVideoQuant	Gets video encoding quant
AdvDVP_SetVideoQuant	Sets video encoding quant
AdvDVP_GetVideoFrameRate	Gets video encoding frame rate
AdvDVP_SetVideoFrameRate	Sets video encoding frame rate
AdvDVP_GetVideoResolution	Gets video encoding resolution
AdvDVP_SetVideoResolution	Sets video encoding resolution
AdvDVP_GetVideoKeyInterval	Gets video encoding key interval
AdvDVP_SetVideoKeyInterval	Sets video encoding key interval
AdvDVP_GetVideoQuality	Gets video encoding quality
AdvDVP_SetVideoQuality	Sets video encoding quality
AdvDVP_GetVideoCompressFormats	Gets the video compression formats list
AdvDVP_GetCurVideoCompressFormat	Gets current video compression format
AdvDVP_GetAudioCompressFormats	Gets the audio compression formats list
AdvDVP_GetCurAudioCompressFormat	Gets current audio compression format
AdvDVP_GetAudioFormat	Gets the compression format of audio
AdvDVP_SetAudioFormat	Sets the compression format of audio

De-interlace control	
AdvDVP_SetDeinterlace	Enables video de-interlace function of not
AdvDVP_GetDeinterlace	Gets the flag of de-interlace function

AVI control	
AdvDVP_CreateAVIFile	Creates AVI file
AdvDVP_WriteAVIFile	Writes video data to AVI file
AdvDVP_CloseAVIFile	Close AVI file
AdvDVP_AudioToAVIFile	Append an audio track to AVI file

“C” exports	
AdvDVP_CreateEncSDKInstance_C	Creates and gets a SDK instance of encoding object
AdvDVP_*XXXX*_C	*XXXX* means all of encoder function names

2.3 Playback APIs

AdvDVP- Series Player Functions Library
Library: DVP7010BPlayer.DLL

Data Type Summary

PlayerRes	The function returns value
EPlayerStatus	The function returns status of playing

Method Summary

Initialize and close	
AdvDVP_CreatePlayerSDKInstance	Creates a SDK instance of Player object

Playback control	
AdvDVP_Openfile	Opens file and initializes player
AdvDVP_CloseFile	Close file that has been opened.
AdvDVP_Play	Plays file that has been opened
AdvDVP_Pause	Pauses or continues
AdvDVP_Stop	Stops playing file
AdvDVP_Fast	Plays file with faster speed
AdvDVP_Slow	Plays file with slower speed
AdvDVP_PlayStep	Plays by single frame
AdvDVP_GetStatus	Gets playback state
AdvDVP_GetCurImage	Gets frame that is rendered
AdvDVP_RegNotifyMsg	Registers message send to player when event occurs
AdvDVP_CheckFileEnd	Checks if file playing is finished
AdvDVP_UpdateVideo	Refreshs the video display

Playback setting	
AdvDVP_GetVideoResolution	Gets video resolution of file
AdvDVP_GetFileTime	Gets total file time
AdvDVP_GetPlayerTime	Gets current file time
AdvDVP_SetPlayPosition	Locates position of file
AdvDVP_GetFileTotalFrames	Gets total frame number of file
AdvDVP_GetPlayedFrames	Gets current frame number of file
AdvDVP_GetPlayRate	Gets current playing rate

“C” exports	
AdvDVP_CreatePlayerSDKInstance_C	Creates and gets a SDK instance of Player object
AdvDVP_*XXXX*_C	*XXXX* means all of player function names

Chapter 3

Functions Description

3.1 Capture APIs

AdvDVP- Series Functions Reference

Library: DVP7010B.DLL

3.1.1 Data Type

Res

Syntax

```
typedef enum tagRes
{
        SUCCEEDED          = 1,
        FAILED             = 0,
        SDKINITFAILED      = -1,
        PARAMERROR         = -2,
        NODEVICES          = -3,
        NOSAMPLE           = -4,
        DEVICENUMERROR     = -5,
        INPUTERROR         = -6,
       //VERIFYHWERROR     = -7,
        NOSUPPORT          = -8,
        CALLBACKERROR      = -9,
        RES_UNKNOWN        = -100
} Res;
```

Description

The function returns value.

3.1.2 Methods

3.1.2.1 Initialize and close

AdvDVP_CreateSDKInstance

Syntax

```
int AdvDVP_CreateSDKInstance(void **pp)
```

Parameters

[OUT] pp: A pointer to the SDK instance.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
PARAMERROR:	Parameter error.

Description

This function creates SDK instance. You can delete the pointer which is carried by the parameter named "pp" to free this SDK instance before you free the DLL handle.

See Also

AdvDVP_InitSDK

AdvDVP_CloseSDK

"C" export

DWORD AdvDVP_CreateSDKInstance_C()

Return a **DWORD** value to indicate the instance of encoding SDK, return NULL meaning function failed.

AdvDVP_InitSDK

Syntax

int AdvDVP_InitSDK()

Parameters

None

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NODEVICES:	No devices found.
SDKINITFAILED:	SDK not initialized.

Description

This function initializes all video capture devices in the system. After initializing each device, the capture status would be set as "STOPPED".

See Also

AdvDVP_CreateSDKInstance

AdvDVP_CloseSDK

AdvDVP_GetNoOfDevices

"C" export

int AdvDVP_InitSDK_C(DWORD dwObj)

AdvDVP_CloseSDK

Syntax

int AdvDVP_CloseSDK(void)

Parameters

None

Return Value

SUCCEEDED:	Function succeeded.
SDKINITFAILED:	SDK not initialized.

Description

This function cleans all instances of capture devices and closes up the SDK.

See Also

AdvDVP_CreateSDKInstance
AdvDVP_InitSDK

"C" export

int AdvDVP_CloseSDK_C(DWORD dwObj)

3.1.2.2 Capture control

AdvDVP_GetNoOfDevices

AdvDVP_GetNoOfAudios

Syntax

int AdvDVP_GetNoOfDevices(int *pNoOfDevs)
int AdvDVP_GetNoOfAudios(int *pNoOfDevs)

Parameters

[OUT] pNoOfDevs: A pointer to get the number of video/audio capture devices.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
PARAMERROR:	Invalid parameter.

Description

This function gets the number of video/audio capture devices in the system.

See Also

AdvDVP_InitSDK
AdvDVP_CloseSDK

"C" export

int AdvDVP_GetNoOfDevices_C(DWORD dwObj, int *pNoOfDevs)
int AdvDVP_GetNoOfAudios_C(DWORD dwObj, int *pNoOfDevs)

AdvDVP_Start

AdvDVP_StartAudio

Syntax

int AdvDVP_Start(int nDevNum, int SwitchingChans, HWND Main, HWND hwndPreview, int nDDraw=0)

int AdvDVP_StartAudio(int nDevNum, int SwitchingChans, HWND Main, HWND hwndPreview)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] SwitchingChans: Single video input or switching between video muxes. Only for video.

0: single channel.

1: two channels (mux0, mux1).

2: three channels (mux0, mux1, mux2).

3: four channels (mux0, mux1, mux2, mux3).

[IN] Main: A main window handle. The handle value can be NULL.

[IN] hwndPreview: A windows handle for display area. This parameter is only valid when the "SwitchingChans)" is zero. When the value of this parameter is NULL, the video will not be rendered.

[IN] nDDraw: Video drawing method. Default is 0. Only for video.

0: using DirectDraw thru render; 1: using DIBDraw; 2: using DirectDraw, SDK create own video surface.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function starts video/audio capturing on a specified capture port. The capture state would be set as "RUNNING" after a successful start. If the channels share frames (i.e. SwitchingChans>0), the video input mux will be set to 0.

See Also

AdvDVP_Stop

AdvDVP_StopAudio

"C" export

int AdvDVP_Start_C(DWORD dwObj, int nDevNum, int SwitchingChans, HWND Main, HWND hwndPreview, int nDDraw)

int AdvDVP_StartAudio_C(DWORD dwObj, int nDevNum, int SwitchingChans, HWND Main, HWND hwndPreview)

AdvDVP_Stop

AdvDVP_StopAudio

Syntax

int AdvDVP_Stop(int nDevNum)

int AdvDVP_StopAudio(int nDevNum)

Parameters

[IN] nDevNum: Specifies the device number.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.

Description

This function stops video/audio capturing on a specified capture port. The capture state would be set as "STOPPED" after a successful stop.

See Also

AdvDVP_Start

AdvDVP_StartAudio

"C" export

int AdvDVP_Stop_C(DWORD dwObj, int nDevNum)

int AdvDVP_StopAudio_C(DWORD dwObj, int nDevNum)

AdvDVP_GetCapState

AdvDVP_GetCapAudioState

Syntax

int AdvDVP_GetCapState(int nDevNum)
int AdvDVP_GetCapAudioState(int nDevNum)

Parameters

[IN] nDevNum: Specifies the device number.

Return Valu

STOPED:	Capture port is stopped.
RUNNING:	Capture port is running.
UNINITIALIZED:	SDK or capture port is not initialized.
UNKNOWNSTATE:	Unknown state.
FAILED:	Function failed.
DEVICENUMERROR:	Invalid device number.

Description

This function gets capture state of a specified capture port.

```
typedef enum {
        STOPPED         = 1,
        RUNNING         = 2,
        UNINITIALIZED   = -1,
        UNKNOWNSTATE    = -2
} CapState;
```

See Also

AdvDVP_Start
AdvDVP_Stop
AdvDVP_StartAudio
AdvDVP_StopAudio

"C" export

int AdvDVP_GetCapState_C(DWORD dwObj, int nDevNum)
int AdvDVP_GetCapAudioState_C(DWORD dwObj, int nDevNum)

AdvDVP_SetNewFrameCallback

Syntax

int AdvDVP_SetNewFrameCallback(int nDevNum, int callback,
int rawF=RAW_DEFAULT)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] callback: Video RAW data callback function.

[IN] rawF: RAW data format.

```
typedef enum
{
    RAW_DEFAULT=0,
    RAW_YUY2,
    RAW_UYVY,
    RAW_RGB16,
    RAW_RGB24,
    RAW_RGB32
} RAWFormat;
```

Callback fumction type:

typedef int (*CAPCALLBACK)(int lParam, int nID, int nDevNum, int VMux, int bufsize, BYTE* buf);

Parameters

[IN] lParam: Reserved.
[IN] nID: Single video input ID or the video mux ID. The value of IDs is showed as the following:

```
#define ID_NEW_FRAME        37810
#define ID_MUX0_NEW_FRAME   37800
#define ID_MUX1_NEW_FRAME   37801
#define ID_MUX2_NEW_FRAME   37802
#define ID_MUX3_NEW_FRAME   37803
```

[IN] nDevNum: Specifies the device number.
[IN] VMux: Specifies the video mux number(0~3).
[IN] bufsize: An integer pointer of the frame buffer size.
[IN] buf: A pointer of the frame buffer.

Return Value

SUCCEEDED:	Function succeeded.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function sets a callback function for SDK. When new frame arrives, messages and frame information will be sent to callback function.

See Also

AdvDVP_Start

"C" export

int AdvDVP_SetNewFrameCallback_C(DWORD dwObj, int nDevNum,
int callback, int rawF=RAW_DEFAULT)

AdvDVP_GetCurFrameBuffer

Syntax

int AdvDVP_GetCurFrameBuffer(int nDevNum, int VMux, long* bufSize, BYTE* buf)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] VMux: Video mux.

[IN/OUT] bufSize: Frame buffer size, in bytes.

[IN] buf: A pointer of frame buffer.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.
NOSAMPLE:	No buffer sample.

Description

This function gets current frame buffer of a specified capture port. Call this function after the AdvDVP_Start, and it should sleep for a little time, such as 500 milliseconds. The default video format is YUYV. To calculate the size of a frame, use the following formula:

```
Buffer Size = Image Height * Image Width * 2;
```

See Also

AdvDVP_GetCurFrameSize

"C" export

int AdvDVP_GetCurFrameBuffer_C(DWORD dwObj, int nDevNum,
int VMux, long* bufSize, BYTE* buf)

AdvDVP_GetCurFrameSize

Syntax

int AdvDVP_GetCurFrameSize(int nDevNum, VideoSize Size,
AnalogVideoFormat avf, int* Width, int* Height)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] Size: The constant of video size

```
typedef enum
{
        SIZE1080        =0,
        SIZE720         =1,
        SIZED1          =2,
        SIZE4CIF        =3,
        SIZEVGA         =4,
        SIZEQVGA        =5,
        SIZECIF         =6,
        SIZESUBQVGA     =7,
        SIZEQCIF        =8
} VideoSize;
```

[IN/OUT] avf: The constant of video format.

```
typedef enum
{
        Video_None          =0,
        Video_NTSC_M        =1,
        Video_NTSC_M_J      =2,
        Video_PAL_B         =3,
        Video_PAL_M         =4,
        Video_PAL_N         =5,
        Video_SECAM_B       =6
} AnalogVideoFormat;
```

[OUT] Width: The pointer to get width of video frame.

[OUT] Height: The pointer to get height of video frame.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function gets the real width and height of video frame.

See Also

AdvDVP_GetCurFrameBuffer

"C" export

int AdvDVP_GetCurFrameSize_C(DWORD dwObj, int nDevNum, VideoSize Size, AnalogVideoFormat avf, int* Width, int* Height)

AdvDVP_SetAudioCallback

Syntax

int AdvDVP_SetAudioCallback(int nDevNum, int callback)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] callback: Audio RAW data callback function.

Callback fumction type:

typedef int (*AUDIOCALLBACK)(double dwSampleTime, int nDevNum, int nAVSyncChannel, int bufsize, BYTE* buf);

Parameters

[IN] dwSampleTime: The timestamp of audio RAW data.

[IN] nID: Single video input ID.

[IN] nDevNum: Specifies the device number.

[IN] nAVSyncChannel: The video channel synchronize with this audio RAW data.

[IN] bufsize: An integer pointer of the audio buffer size.

[IN] buf: A BYTE pointer of the audio buffer.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function sets a callback function for SDK. When new audio data arrives, messages and frame information will be sent to callback function.

See Also

AdvDVP_SetNewFrameCallback

"C" export

int AdvDVP_SetAudioCallback_C(DWORD dwObj, int nDevNum, int callback)

AdvDVP_GetNoOfChannels

Syntax

int AdvDVP_GetNoOfChannels(int *pNoOfChannels)

Parameters

[OUT] pNoOfChannels: A pointer to get the number of video channels.

Return Value

SUCCEEDED: Function succeeded.
SDKINITFAILED: SDK not initialized.
PARAMERROR: Invalid parameter.

Description

This function gets the number of total video channels in the system.

See Also

AdvDVP_GetNoOfDevices

"C" export

int AdvDVP_GetNoOfChannels_C(DWORD dwObj, int *pNoOfChannels)

AdvDVP_GetCurChannel

Syntax

int AdvDVP_GetCurChannel(int nDevNum, int *pnCurChannel)

Parameters

[IN] nDevNum: Specifies the device number.
[OUT] pnCurChannel: A pointer to get the number of video channel relative with nDevNum.

Return Value

SUCCEEDED: Function succeeded.
SDKINITFAILED: SDK not initialized.
PARAMERROR: Invalid parameter.

Description

This function gets the number of video channel, which relative with nDevNum parameter, in the system.

See Also

AdvDVP_GetNoOfChannels

"C" export

int AdvDVP_GetCurChannel_C(DWORD dwObj, int nDevNum, int* pnChannel)

AdvDVP_ChannelToDevice

Syntax

int AdvDVP_ChannelToDevice(int nChannel, int* pnDevNum)

Parameters

[IN] nChannel: Specifies the video channel number.

[OUT] pnDevNum: A pointer to get the number of video device.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
PARAMERROR:	Invalid parameter.

Description

This function gets the number of video device, which relative with nChannel parameter, in the system.

See Also

AdvDVP_GetCurChannel

"C" export

int AdvDVP_ChannelToDevice_C(DWORD dwObj, int nChannel, int *pnDevNum)

3.1.2.3 Capture setting

AdvDVP_GetVideoFormat

Syntax
int AdvDVP_GetVideoFormat(int nDevNum, AnalogVideoFormat* vFormat)

Parameters
[IN] nDevNum: Specifies the device numbeR.
[OUT] Vformat: A pointer to get video format.

```
typedef enum  tagAnalogVideoFormat
{
        Video_None = 0x00000000,
        Video_NTSC_M = 0x00000001,
        Video_NTSC_M_J = 0x00000002,
        Video_PAL_B = 0x00000010,
        Video_PAL_M = 0x00000200,
        Video_PAL_N = 0x00000400,
        Video_SECAM_B = 0x00001000
} AnalogVideoFormat;
```

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description
This function gets video input format of a specified capture port.

See Also
AdvDVP_SetVideoFormat

"C" export

```
int AdvDVP_GetVideoFormat_C(DWORD dwObj, int nDevNum,
                            AnalogVideoFormat* vFormat)
```

AdvDVP_SetVideoFormat

Syntax

int AdvDVP_SetVideoFormat(int nDevNum, AnalogVideoFormat vFormat)

Parameters

[IN] nDevNum: Specifies the port device number.

[IN] Vformat: Video format:

```
typedef enum tagAnalogVideoFormat
{
        Video_None = 0x00000000,
        Video_NTSC_M = 0x00000001,
        Video_NTSC_M_J = 0x00000002,
        Video_PAL_B = 0x00000010,
        Video_PAL_M = 0x00000200,
        Video_PAL_N = 0x00000400,
        Video_SECAM_B = 0x00001000
} AnalogVideoFormat;
```

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.

Description

This function sets video input format of a specified capture port. This function should be called before "AdvDVP_Start".

See Also

AdvDVP_GetVideoFormat

"C" export

int AdvDVP_SetVideoFormat_C(DWORD dwObj, int nDevNum,
AnalogVideoFormat vFormat)

AdvDVP_GetFrameRate

Syntax

int AdvDVP_GetFrameRate(int nDevNum, int *FrameRate)

Parameters

[IN] nDevNum: Specifies the device number.
[OUT] FrameRate: A pointer to get video frame rate.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function gets frame rate of a specified capture port.

See Also

AdvDVP_SetFrameRate

"C" export

int AdvDVP_GetFrameRate_C(DWORD dwObj, int nDevNum, int *FrameRate)

AdvDVP_SetFrameRate

Syntax

int AdvDVP_SetFrameRate(int nDevNum , int SwitchingChans, int FrameRate)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] SwitchingChans: Single video input or switching between video muxes(0~3).

0: single channel.

1: two channels (mux0, mux1).

2: three channels (mux0, mux1, mux2).

3: four channels (mux0, mux1, mux2, mux3).

[IN] FrameRate: A value to set frame rate. (0<FrameRate<=30, Default value is 30)

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function sets frame rate of a specified capture port. This function should be called before "AdvDVP_Start". If the channels share frames (i.e. SwitchingChans>0), the frame rate must be set to 30. Otherwise, the function will return PARAMERROR.

See Also

AdvDVP_GetFrameRate

"C" export

int AdvDVP_SetFrameRate_C(DWORD dwObj, int nDevNum, int Switching,
int FrameRate)

AdvDVP_GetResolution

Syntax

int AdvDVP_GetResolution(int nDevNum, VideoSize *Size)

Parameters

[IN] nDevNum: Specifies the device number.

[OUT] Size: A pointer to get video resolution.

```
typedef enum
{
        SIZE1080        =0,
        SIZE720         =1,
        SIZED1          =2,
        SIZE4CIF        =3,
        SIZEVGA         =4,
        SIZEQVGA        =5,
        SIZECIF         =6,
        SIZESUBQVGA     =7,
        SIZEQCIF        =8
} VideoSize;
```

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function gets video resolution of a specified capture port.

See Also

AdvDVP_SetResolution

"C" export

int AdvDVP_GetResolution_C(DWORD dwObj, int nDevNum, VideoSize *Size)

AdvDVP_SetResolution

Syntax

int AdvDVP_SetResolution(int nDevNum, VideoSize Size)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] Size: A value to set video resolution.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.

Description

This function sets video resolution of a specified capture port. This function should be called before "AdvDVP_Start".

See Also

AdvDVP_GetResolution

"C" export

int AdvDVP_SetResolution_C(DWORD dwObj, int nDevNum, VideoSize Size)

AdvDVP_GetVideoInput

Syntax

int AdvDVP_GetVideoInput(int nDevNum, int* pInput)

Parameters

[IN] nDevNum: Specifies the device number.

[OUT] pInput: A pointer to get video input mux.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	The "pInput" parameter is invalid.

Description

This function gets video input mux of a specified capture port. It returns "FAILED" when argument "SwitchingChans" of AdvDVP_Start was set to nonzero. And, the video input mux will be set to 0 automatically when argument "SwitchingChans" of AdvDVP_Start was set to nonzero.

See Also

AdvDVP_SetVideoInput

"C" export

int AdvDVP_GetVideoInput_C(DWORD dwObj, int nDevNum, int *pInput)

AdvDVP_SetVideoInput

Syntax

int AdvDVP_SetVideoInput(int nDevNum, int nInput)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] nInput: A value to set video input mux(0~3).

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid video input.

Description

This function sets video input mux of a specified capture port. It returns "FAILED" when argument "SwitchingChans" of AdvDVP_Start was set to nonzero. And, the video input mux will be set to 0 automatically when argument "SwitchingChans" of AdvDVP_Start was set to nonzero.

See Also

AdvDVP_GetVideoInput

"C" export

int AdvDVP_SetVideoInput_C(DWORD dwObj, int nDevNum, int nInput)

AdvDVP_GetVideoSwitching

Syntax

int AdvDVP_GetVideoSwitching(int nDevNum, int* pnVideoSwitch, int *pnVideoMUX)

Parameters

[IN] nDevNum: Specifies the device number.

[OUT] pnVideoSwitch: A pointer to get the number of video switching channel.

[OUT] pnVideoMUX: A pointer to get the number of video mux.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	The parameter is invalid.

Description

This function gets video switching channel and mux of a specified capture port.

See Also

AdvDVP_GetVideoInput

"C" export

int AdvDVP_GetVideoSwitching_C(DWORD dwObj, int nDevNum,
int *pnVideoSwitch, int *pnVideoMUX)

AdvDVP_GetMappingResolution

Syntax

int AdvDVP_GetMappingResolution(int nDevNum, VideoSize Size,
AnalogVideoFormat avf, int *Width, int *Height)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] Size: A value to set video resolution.

[IN] avf: The constant of video format.

[OUT] Width: A pointer to get the width of video resolution.

[OUT] Height: A pointer to get the height of video resolution.

Return Value

SUCCEEDED:	Function scceeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid video input.

Description

This function gets video real width and height, which relative with the parameter "Size" and "avf".

See Also

AdvDVP_GetResolution

"C" export

int AdvDVP_GetMappingResolution_C(DWORD dwObj, int nDevNum,
VideoSize Size, AnalogVideoFormat avf,
int *Width, int *Height)

AdvDVP_GetAudioFormat

Syntax

int AdvDVP_GetAudioFormat(int nDevNum, int *pnChannel, int *pnBitsPerSample, int *pnSamplePerSec)

Parameters

[IN] nDevNum: Specifies the device number.

[OUT] pnChannel: A pointer to get the channel number of audio.

[OUT] pnBitsPerSample: A pointer to get the bits number of audio sample.

[OUT] pnSamplePerSec: A pointer to get the audio sampling number for each second.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function gets audio input format of a specified capture port.

See Also

AdvDVP_SetAudioFormat

"C" export

int AdvDVP_GetAudioFormat_C(DWORD dwObj, int nDevNum, int *pnChannel, int *pnBitsPerSample, int *pnSamplePerSec)

AdvDVP_SetAudioFormat

Syntax

int AdvDVP_SetAudioFormat(int nDevNum, int nChannel, int nBitsPerSample,
int nSamplePerSec)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] nChannel: The channel number of audio. 1 is mono, 2 is stereo.

[IN] nBitsPerSample: The bits number of audio sample. Default is 16.

[IN] nSamplePerSec: The audio sampling number for each second. Default is 8000.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.

Description

This function sets audio input format of a specified capture port.

See Also

AdvDVP_GetAudioFormat

"C" export

int AdvDVP_SetAudioFormat_C(DWORD dwObj, int DevNum, int nChannel,
int nBitsPerSample, int nSamplePerSec)

AdvDVP_GetAudioVolume

Syntax

int AdvDVP_GetAudioVolume(int nDevNum, int *pnVolume)

Parameters

[IN] nDevNum: Specifies the device number.
[OUT] pnVolume: A pointer to get audio volume.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function gets the audio volume of a specified capture port.

See Also

AdvDVP_SetAudioVolume

"C" export

int AdvDVP_GetAuidoVolume_C(DWORD dwObj, int nDevNum, int *pnVolume)

AdvDVP_SetAudioVolume

Syntax

int AdvDVP_SetAudioVolume(int nDevNum, int nVolume)

Parameters

[IN] nDevNum: Specifies the device number.
[IN] nVolume: The volume of audio.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.

Description

This function sets the audio volume of a specified capture port.

See Also

AdvDVP_GetAudioVolume

"C" export

int AdvDVP_SetAuidoVolume_C(DWORD dwObj, int nDevNum, int nVolume)

AdvDVP_SetAVSyncChannel

Syntax

int AdvDVP_SetAVSyncChannel(int nDevNum, int nAVSyncChannel)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] nAVSyncChannel: The channel number of video.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.

Description

This function sets the audio channel synchronize with video channel. The "nAVSync-Channel" will pass to audio callback function.

See Also

"C" export

int AdvDVP_SetAVSyncChannel_C(DWORD dwObj, int nDevNum, int nAVSyncChannel)

3.1.2.4Sensor control

AdvDVP_GetBrightness

Syntax

AdvDVP_GetBrightness(int nDevNum, int nInput, long *lpValue)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] nInput: Specifies the video input mux(-1~3). This value must be set to -1 when there are no switching channels.

[OUT] lpValue: A long pointer to get brightness value.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
INPUTERROR:	Invalid video input mux.
PARAMERROR:	The "lpValue" parameter is invalid.

Description

This function gets brightness value of a specified capture port.

See Also

AdvDVP_SetBrightness

"C" export

int AdvDVP_GetBrightness_C(ValueDWORD dwObj, int nDevNum,
int nInput long *lpValue)

AdvDVP_SetBrightness

Syntax

int AdvDVP_SetBrightness(int nDevNum , int nInput, long lValue)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] nInput: Specifies the video input mux(-1~3). This value must be set to -1 when no switching channels.

[IN] lValue: A value to set brightness(0~100).

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
INPUTERROR:	Invalid video input mux.
PARAMERROR:	Invalid brightness value.

Description

This function sets brightness value of a specified capture port.

See Also

AdvDVP_GetBrightness

"C" export

int AdvDVP_SetBrightness_C(DWORD dwObj, int nDevNum, int nInput
long lVaue)

AdvDVP_GetContrast

Syntax

int AdvDVP_GetContrast(int nDevNum, int nInput, long *lpValue)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] nInput: Specifies the video input mux(-1~3). This value must be set to -1 when no switching channels.

[OUT] lpValue: A long pointer to get contrast value.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
INPUTERROR:	Invalid video input mux.

PARAMERROR: The "lpValue" parameter is invalid.

Description

This function gets contrast value of a specified capture port.

See Also

AdvDVP_SetContrast

"C" export

int AdvDVP_GetContrast_C(DWORD dwObj, int nDevNum, int nInput
long *lpValue)

AdvDVP_SetContrast

Syntax

int AdvDVP_SetContrast(int nDevNum, int nInput, long lValue)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] nInput: Specifies the video input mux(-1~3). This value must be set to -1 when there are no switching channels.

[IN] lValue: A value to set contrast(0~100).

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device.
INPUTERROR:	Invalid video input mux.
PARAMERROR:	Invalid contrast value.

Description

This function sets contrast value of a specified capture port.

See Also

AdvDVP_GetContrast

"C" export

int AdvDVP_SetContrast_C(DWORD dwObj, int nDevNum, int nInput,
long lValue)

AdvDVP_GetHue

Syntax

int AdvDVP_GetHue(int nDevNum, int nInput, long *lpValue)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] nInput: Specifies the video input mux(-1~3). This value must be set to -1 when there are no switching channels.

[OUT] lpValue: A long pointer to get hue value.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
INPUTERROR:	Invalid video input mux.
PARAMERROR:	The "lpValue" parameter is invalid.

Description

This function gets hue value of a specified capture port.

See Also

AdvDVP_SetHue

"C" export

int AdvDVP_GetHue_C(DWORD dwObj, int nDevNum, int nInput
long *lpValue)

AdvDVP_SetHue

Syntax

int AdvDVP_SetHue(int nDevNum, int nInput, long lValue)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] nInput: Specifies the video input mux(-1~3). This value must be set to -1 when there are no switching channels.

[IN] lValue: A value to set hue(0~100).

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
INPUTERROR:	Invalid video input mux.

PARAMERROR:	Invalid hue value.

Description

This function sets hue value of a specified capture port.

See Also

AdvDVP_GetHue

"C" export

int AdvDVP_SetHue_C(DWORD dwObj, int nDevNum, int nInput, long lValue)

AdvDVP_GetSaturation

Syntax

int AdvDVP_GetSaturation(int nDevNum, int nInput, long *lpValue)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] nInput: Specifies the video input mux(-1~3). This value must be set to -1 when there are no switching channels.

[OUT] lpValue: A long pointer to get saturation value.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	The "lpValue" parameter is invalid.
INPUTERROR:	Invalid video input mux.

Description

This function gets saturation value of a specified capture port.

See Also

AdvDVP_SetSaturation

"C" export

int AdvDVP_GetSaturation_C(DWORD dwObj, int nDevNum, int nInput, long *lpValue)

AdvDVP_SetSaturation

Syntax

int AdvDVP_SetSaturation(int nDevNum , int nInput, long lValue)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] nInput: Specifies the video input mux(-1~3). This value must be set to -1 when there are no switching channels.

[IN] lValue: A value to set saturation(0~100).

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
INPUTERROR:	Invalid video input mux.
PARAMERROR:	Invalid saturation value.

Description

This function sets saturation value of a specified capture port.

See Also

AdvDVP_GetSaturation

"C" export

int AdvDVP_SetSaturation_C(DWORD dwObj, int nDevNum, int nInput, long lValue)

3.1.2.5GPIO control

AdvDVP_GPIOGetData

Syntax

int AdvDVP_GPIOGetData(int nDevNum, int nDINum, BOOL* pValue)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] nDINum: Specifies the digital input number.

[OUT] pValue: A pointer to get the value of the specified digital input.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function gets the value of the specified digital input.

See Also

AdvDVP_GPIOSetData

"C" export

int AdvDVP_GPIOGetData_C(DWORD dwObj, int nDevNum, int nDINum,
BOOL *pValue)

AdvDVP_GPIOSetData

Syntax

int AdvDVP_GPIOSetData(int nDevNum, int nDONum, BOOL bValue)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] nDONum: Specifies the digital output number.

[IN] bValue: A value to set the value of the specified digital output.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.

Description

This function sets the value of the specified digital output.

See Also

AdvDVP_GPIOGetData

"C" export

int AdvDVP_GPIOSetData_C(DWORD dwObj, int nDevNum, int nDONum, BOOL bValue)

3.1.2.6 De-interlace control

AdvDVP_GetDeinterlace

Syntax
BOOL AdvDVP_GetDeinterlace()

Parameters
None.

Return Value
TRUE: De-interlace is enabled.
FALSE: De-interlace is disabled.

Description
This function retrieves de-interlace state.

See Also
AdvDVP_SetDeinterlace

"C" export
BOOL AdvDVP_GetDeinterlace_C(DWORD dwObj)

AdvDVP_SetDeinterlace

Syntax
void AdvDVP_SetDeinterlace(BOOL bEnable)

Parameters
[IN] bEnable: TRUE for Enable; False for Disable.

Return Value
None.

Description
This function enables or disables the de-interlace function.

See Also
AdvDVP_GetDeinterlace

"C" export
int AdvDVP_SetDeinterlace_C(DWORD dwObj, BOOL bEnable)

3.1.2.7 Text OSD

AdvDVP_GetNumOfOSD

Syntax

int AdvDVP_GetNumOfOSD(int nDevNum, int *pNumOfOSD)

Parameters

[IN] nDevNum: Specifies the device number.

[OUT] pNumOfOSD: A pointer to get the number of supported OSD.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function gets the number of supported text OSD on the video capture port.

See Also

AdvDVP_SetTextOSDEx

AdvDVP_ClearTextOSDEx

"C" export

int AdvDVP_GetNumOfOSD_C(DWORD dwObj, int nDevNum, int *pNumOfOSD)

AdvDVP_SetTextOSDEx

Syntax

int AdvDVP_SetTextOSDEx(int nDevNum, int nOSDNum, DVPOSD *pOSD)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] nOSDNum: Specifies the OSD number.

[IN] pOSD: The pointer of 'DVPOSD' structure.

```
typedef struct
{
        int             nXpos;
        int             nYpos;
        int             nOSDLength;
        char            chTextOSD[OSDLENGTH];
        OSDFontMode     osdFontMode;
        int             nTranslucent;// 0 ~ 255
        int             nLocation;
        int             nDTSFormat;
```

} DVPOSD;

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function set / show text OSD display.

See Also

AdvDVP_GetNumOfOSD
AdvDVP_ClearTextOSDEx

"C" export

int AdvDVP_SetTextOSDEx_C(DWORD dwObj, int nDevNum, int nOSDNum, DVPOSD *pOSD)

AdvDVP_ClearTextOSDEx

Syntax

int AdvDVP_ClearTextOSDEx(int nDevNum, int nOSDNum)

Parameters

[IN] nDevNum: Specifies the device number.
[IN] nOSDNum: Specifies the OSD number.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.

Description

This function clear / hide text OSD display. If nOSDNum is -1, means clear all of text OSD.

See Also

AdvDVP_GetNumOfOSD
AdvDVP_SetTextOSDEx

"C" export

int AdvDVP_ClearTextOSDEx_C(DWORD dwObj, int nDevNum, int nOSDNum)

3.1.2.8 Compression control

AdvDVP_StartCompress

Syntax

int AdvDVP_StartCompress(int nDevNum)

Parameters

[IN] nDevNum: Specifies the device number.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
CALLBACKERROR:	Call back function entry point is not available.

Description

This function starts video compression on a specified capture port. The compression state would be set as "RUNNING" after a successful start.

See Also

AdvDVP_StopCompress

"C" export

int AdvDVP_StartCompress_C(DWORD dwObj, int nDevNum)

AdvDVP_StopCompress

Syntax

int AdvDVP_StopCompress(int nDevNum)

Parameters

[IN] nDevNum: Specifies the device number.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.

Description

This function stops compression capturing on a specified capture port. The compression state would be set as "STOPPED" after a successful stop.

See Also

AdvDVP_StartCompress

"C" export

int AdvDVP_StopCompress_C(DWORD dwObj, int nDevNum)

AdvDVP_GetCompressState

Syntax

int AdvDVP_GetCompressState(int nDevNum)

Parameters

[IN] nDevNum: Specifies the device number.

Return Value

STOPED:	Capture port is stopped.
RUNNING:	Capture port is running.
UNINITIALIZED:	SDK or capture port is not initialized.
UNKNOWNSTATE:	Unknown state.
FAILED:	Function failed.
DEVICENUMERROR:	nvalid device number.

Description

This function gets compression state of a specified capture port.

```
typedef enum {
        STOPPED         = 1,
        RUNNING         = 2,
        UNINITIALIZED   = -1,
        UNKNOWNSTATE    = -2
} CapState;
```

See Also

AdvDVP_StartCompress
AdvDVP_StopCompress

"C" export

int AdvDVP_GetCompressState_C(DWORD dwObj, int nDevNum)

AdvDVP_RestartCompress

Syntax

int AdvDVP_RestartCompress(int nDevNum)

Parameters

[IN] nDevNum: Specifies the device number.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
CALLBACKERROR:	Call back function entry point is not available.

Description

This function restarts video compression on a specified capture port. The compression state would be set as "RUNNING" after a successful start.

See Also

AdvDVP_StartCompress

"C" export

int AdvDVP_RestartCompress_C(DWORD dwObj, int nDevNum)

AdvDVP_SetCompressCallback

Syntax

int AdvDVP_SetCompressCallback(int nDevNum, int callback)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] callback: Video compression data callback function.

Callback fumction type:

typedef int (*COMPCALLBACK)(double dwSampleTime, int nID, int nDevNum, int nAVSyncChannel, int bufsize, BYTE* buf);

Parameters

[IN] dwSampleTime: The timestamp of audio RAW data.

[IN] nID: Single video input ID.

The value of IDs is showed as the following:

#define ID_BEGIN_COMPRESS	38800
#define ID_FRAME_COMPRESS	38801
#define ID_END_COMPRESS	38802
#define ID_KEYFRAME_COMPRESS	38803

[IN] nDevNum: Specifies the device number.
[IN] nAVSyncChannel: The video channel synchronize with this audio RAW data.
[IN] bufsize: An integer pointer of the audio buffer size, in bytes.
[IN] buf: A pointer of the audio buffer.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function sets a callback function for SDK. When new compression data arrives, messages and frame information will be sent to callback function.

See Also

AdvDVP_StartCompress

"C" export

int AdvDVP_SetCompressCallback_C(DWORD dwObj, int nDevNum, int callback)

3.1.2.9 Compression setting

AdvDVP_GetVideoCompressFormats

Syntax

int AdvDVP_GetVideoCompressFormats(COMPRESS_FORMAT_STRUCT *pCompressFormats, int nMaxFormats)

Parameters

[IN] pCompressFormats: A pointer to the COMPRESS_FORMAT_STRUCT structure array buffer.
[IN] nMaxFormats: The number of COMPRESS_FORMAT_STRUCT structure array buffer.

Return Value

Retrieves all of compression formats list, and returns the number of compression formats. If function failed, or capture-only video channel, -1 will be returned.

Description

This function gets all of the available video compression formats of hardware encoding.

See Also

AdvDVP_GetCurVideoCompressFormat
AdvDVP_SetVideoCompressFormat

"C" export

```
int AdvDVP_GetVideoCompressFormats_C(DWORD dwObj,
                                     COMPRESS_FORMAT_STRUCT *pCompressFormats,
                                     int nMaxFormats)
```

AdvDVP_GetCurVideoCompressFormat

Syntax

```
int AdvDVP_GetCurVideoCompressFormat(int nDevNum,
                                     VideoCompressFormat *pVCFormat,
                                     int *pnWidth, int *pnHeight)
```

Parameters

[IN] nDevNum: Specifies the capture device number.

[OUT] pVCFormat: A pointer to get the video compression format.

[OUT] pnWidth:A pointer to get the width of video frame.

[OUT] pnHeight: A pointer to get the height of video frame.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK is not initialized successfully.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function gets the video compression format of hardware encoding.

See Also

AdvDVP_GetVideoCompressFormats

AdvDVP_SetVideoCompressFormat

"C" export

```
int AdvDVP_GetCurVideoCompressFormat_C(DWORD dwObj, int nDevNum,
                                       VideoCompressFormat *pVCFormat,
                                       int *pnWidth, int *pnHeight)
```

AdvDVP_SetVideoCompressFormat

Syntax

int AdvDVP_SetVideoCompressFormat(int nDevNum,
VideoCompressFormat VCFormats,DWORD dwRsvd)

Parameters

[IN] nDevNum: Specifies the capture device number.

[IN] pVCFormat: A pointer to get the video compression format.

[IN] dwRsvd: Reserved.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK is not initialized successfully.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function sets the video compression format to capture device.

See Also

AdvDVP_GetCurVideoCompressFormat

"C" export

int AdvDVP_SetVideoCompressFormat_C(DWORD dwObj, int nDevNum,
VideoCompressFormat VCFormats, DWORD dwRsvd)

AdvDVP_IsSupportCompress

Syntax

int AdvDVP_IsSupportCompress(int nDevNum, int *pIsSupport)

Parameters

[IN] nDevNum: Specifies the capture device number.

[IN] pIsSupport: The pointer to get the flag if supports hardware compression or not.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
SDKINITFAILED:	SDK is not initialized successfully.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function returns 1 meaning hardware compression is supported, otherwise returns 0.

See Also

AdvDVP_GetVideoCompressFormats

"C" export

int AdvDVP_IsSupportCompress_C(DWORD dwObj, int *pIsSupport)

AdvDVP_GetCompressAttr

Syntax

int AdvDVP_GetCompressAttr(int nDevNum, int nAttrId, int *pnVal)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] nAttrId: Specifies the constant of compression attribute.

```
typedef enum
{
        COMP_GOP=0,
        COMP_QUALITY,
        COMP_BITRATEMODE,
        COMP_BITRATE,
        COMP_QPSTEP,
        COMP_PEAKBITRATE,
        COMP_TROUGHQUALITY,
        COMP_RESOLUTION,
        COMP_SKIPFRAMERATE,
        COMP_AVGFRAMERATE,
        COMP_BFRAME,
        COMP_PROFILE,
        COMP_ASPECRATIO,
        COMP_QUANT=100
} CompressAttr;
```

[OUT] pnVal: A pointer to get value.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function gets the compression attribute value of a specified capture port.

See Also

AdvDVP_SetCompressAttr

"C" export

int AdvDVP_GetCompressAttr_C(DWORD dwObj, int nDevNum, int nAttrId,
int *pnVal)

AdvDVP_SetCompressAttr

Syntax

int AdvDVP_SetCompressAttr(int nDevNum , int nAttrId, int nVal)

Parameters

[IN] nDevNum: Specifies the device number.

[IN] nAttrId: Specifies the constant of compression attribute.

[IN] nVal: A value to set the attribute of compression.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK not initialized.
DEVICENUMERROR:	Invalid device number.

Description

This function sets compression attribute value of a specified capture port.

See Also

AdvDVP_GetCompressAttr

"C" export

int AdvDVP_SetCompressAttr_C(DWORD dwObj, int nDevNum,
int nAttrId, int nVal)

3.1.2.10 Misellaneous

AdvDVP_IsVideoPresent

Syntax

int AdvDVP_IsVideoPresent(int nDevNum, BOOL *VPresent)

Parameters

[IN] nDevNum: Specifies the capture device number.
[OUT] VPresent: The pointer to get the boolean value.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK is not initialized successfully.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function gets the boolean value of video input if present or not.

See Also

"C" export

int AdvDVP_IsVideoPresent_C(DWORD dwObj, int nDevNum, BOOL *VPresent)

AdvDVP_SetVideoZoom

Syntax

int AdvDVP_SetVideoZoom(int nDevNum, POINT *pt, int nZoom)

Parameters

[IN] nDevNum: Specifies the capture device number.
[IN] pt: The pointer of POINT structure.
[IN] nZoom: The value of Zoom In/Out. The range is 0.0 ~ 10.0.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK is not initialized successfully.
DEVICENUMERROR:	Invalid device number.
PARAMERROR:	Invalid parameter.

Description

This function enlarges or shrink the video.

See Also

"C" export

int AdvDVP_SetVideoZoom_C(DWORD dwObj, int nDevNum, POINT *pt,
int nZoom)

AdvDVP_EnableWatchDog

Syntax

int AdvDVP_EnableWatchDog(int nDevNum, int nDuration)

Parameters

[IN] nDevNum: Specifies the capture device number.

[IN] nDuration: The time for count up, by milliseconds.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK is not initialized successfully.
DEVICENUMERROR:	Invalid device number.

Description

This function enables the watchdog detection, and sets the duration for count up to reset system.

See Also

AdvDVP_ResetWatchDog

"C" export

int AdvDVP_EnableWatchDog_C(DWORD dwObj, int nDevNum,
int nDuration)

AdvDVP_ResetWatchDog

Syntax

int AdvDVP_ResetWatchDog(int nDevNum)

Parameters

[IN] nDevNum: Specifies the capture device number.

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK is not initialized successfully.
DEVICENUMERROR:	Invalid device number.

Description

This function resets and clears the counter of watchdog.

See Also

AdvDVP_EnableWatchDog

"C" export

int AdvDVP_ResetWatchDog_C(DWORD dwObj, int nDevNum)

AdvDVP_IsSupportVideoFormat

Syntax

int AdvDVP_IsSupportVideoFormat(int nDevNum, AnalogVideoFormat vFormat)

Parameters

[IN] nDevNum: Specifies the capture device number.

[IN] vFormat: The constant of video format.

```
typedef enum  tagAnalogVideoFormat
{
        Video_None          = 0x00000000,
        Video_NTSC_M        = 0x00000001,
        Video_NTSC_M_J      = 0x00000002,
        Video_PAL_B         = 0x00000010,
        Video_PAL_M         = 0x00000200,
        Video_PAL_N         = 0x00000400,
        Video_SECAM_B       = 0x00001000
} AnalogVideoFormat;
```

Return Value

SUCCEEDED:	Function succeeded.
FAILED:	Function failed.
NOSUPPORT:	Function not supported.
SDKINITFAILED:	SDK is not initialized successfully.
DEVICENUMERROR:	Invalid device number.

Description

This function checks if the video port supports such video format or not.

See Also

"C" export

int AdvDVP_IsSupportVideoFormat_C(DWORD dwObj, int nDevNum,
AnalogVideoFormat vFormat)

3.2 Encoding APIs

AdvDVP- Series Functions Reference
Library: DVP7010BEnc.DLL

3.2.1 Data Type

EncRes

Syntax

```
typedef enum tagRes
{
        ENC_SUCCEEDED       = 1,
        ENC_FAILED          = 0,
        ENC_SDKINITFAILED   = -1,
        ENC_ENCINITFAILED   = -2,
        ENC_PARAMERROR      = -3,
        ENC_ENCNUMERROR     = -4,
        ENC_BUFFERFULL      = -5
        ENC_BUSY            = -6
} EncRes;
```

Description

The function returns value.

3.2.2 Callback

PSTREAMREADBEGIN

Syntax

void (*PSTREAMREADBEGIN)(int nEncNum)

Parameters

[IN] nEncNum: Specifies the encoder number.

Return Value

None.

Description

The pointer to the Stream Read Begin callback function will be called when the video stream read process begins.

See Also

STREAMREAD_STRUCT

PSTREAMREADPROC

Syntax

void (*PSTREAMREADPROC)(int nEncNum, LPVOID pStreamBuf, long lBufSize, DWORD dwCompFlags)

Parameters

[IN] nEncNum: Specifies the encoder number.

[IN] pStreamBuf: A pointer to the data buffer that stores an encoded video frame.

[IN] lBufSize: Specifies the size of the encoded video frame.

[IN] dwCompFlags: Specifies if this encoded video frame is I-frame.
The AVIIF_KEYFRAME value means the frame is I-frame.

```
#define AVIIF_KEYFRAME 0x00000010L
```

Return Value

None.

Description

The pointer to the Stream Read Process callback function will be called after every video frame is encoded. The user can use this function to get every encoded video frame.

See Also

STREAMREAD_STRUCT

PSTREAMREADEND

Syntax

void (*PSTREAMREADEND)(int nEncNum)

Parameters

[IN] nEncNum: Specifies the encoder number.

Return Value

None.

Description

The pointer to the Stream Read End callback function will be called when the video stream read process is finished.

See Also

STREAMREAD_STRUCT

3.2.3 Structures

STREAMREAD_STRUCT structure

Syntax

```
typedef struct
{
        void (*PSTREAMREADBEGIN)(int nEncNum);
        void (*PSTREAMREADPROC)(int nEncNum, LPVOID pStreamBuf, long
                                lBufSize, DWORD dwCompFlags);
        void (*PSTREAMREADEND)(int nEncNum);
}STREAMREAD_STRUCT;
```

Parameters

PSTREAMREADBEGIN: The pointer to the Stream Read Begin callback function will be called when begins the video stream read process.

PSTREAMREADPROC: The pointer to the Stream Read Process callback function will be called after every video frame is encoded.

PSTREAMREADEND: The pointer to the Stream Read End callback function will be called when the video stream read process is finished.

Description

This structure stores the Stream Read callback function pointers.

See Also

PSTREAMREADBEGIN

PSTREAMREADPROC

PSTREAMREADEND

AdvDVP_SetStreamReadCB

COMPRESS_FORMAT_STRUCT structure

Syntax

```
typedef struct
{
        DWORD   fccHandler;
        DWORD   dwCompressFormat;
        DWORD   dwEncProcessor;
        char    szName[16];
}STREAMREAD_STRUCT;
```

Parameters

fccHandle: The pointer to the Stream Read Begin callback function will be called when begins the video stream read process.

dwCompressFormat: The pointer to the Stream Read Process callback function will be called after every video frame is encoded.

```
typedef enum
{
        VID_COMP_NONE           = -1,
        VID_COMP_RMP4           = 0,    // S/W compression
        VID_COMP_MPEG4_DIVX     = 1,
        VID_COMP_MPEG4_XVID     = 2,
        VID_COMP_X264           = 3,
        VID_COMP_MPG4_SW        = 4,
        VID_COMP_H264_SW        = 5,
        VID_COMP_HW_COMPRESS    = 10,
        VID_COMP_JPEG           = 10,   // The follwing is H/W compression
        VID_COMP_MPEG2          = 11,
        VID_COMP_MPEG4          = 12,
        VID_COMP_H264_BaseLine  = 13,
        VID_COMP_H264_MainProfile   = 14,   // CABAC
        VID_COMP_H264_MainProfile2  = 15,   // CALVC
        VID_COMP_H264_HighProfile   = 16,   // CABAC
        VID_COMP_H264_HighProfile2  = 17,   // CALVC
} VideoCompressFormat;

typedef enum
{
        AUD_COMP_PCM        = 0,
        AUD_COMP_ALAW       = 1,
        AUD_COMP_MULAW      = 2,
        AUD_COMP_AC3        = 3,
} AudioCompressFormat;
```

dwEncProcessor: The pointer to the Stream Read End callback function will be called when the video stream read process is finished.

szName: The name of compression format.

Description

This structure stores the compression format of video or audio.

See Also

3.2.4 Methods

3.2.4.1 Initialize and close

AdvDVP_CreateEncSDKInstance

Syntax

int AdvDVP_CreateEncSDKInstance (void **pp)

Parameters

[OUT] pp: A pointer to the encoding SDK instance.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_PARAMERROR:	Parameter error.

Description

This function creates the encoding SDK instance. You can delete the pointer which is carried by the parameter named "pp" to free this SDK instance before you free the DLL handle.

"C" export

DWORD AdvDVP_CreateEncSDKInstance_C()

Return a DWORD value to indicate the instance of encoding SDK, return NULL meaning function failed.

AdvDVP_InitSDK

Syntax

int AdvDVP_InitSDK(void)

Parameters

None.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.

Description

This function initializes all parameters of the SDK in the system.

See Also

AdvDVP_CloseSDK

"C" export
int AdvDVP_InitSDK_C(DWORD dwObj)

AdvDVP_CloseSDK

Syntax
int AdvDVP_CloseSDK(void)

Parameters
None.

Return Value
ENC_SUCCEEDED: Function succeeded.
ENC_SDKINITFAILED: SDK is not initialized successfully.

Description
This function cleans all parameters of the SDK and closes up the SDK.

See Also
AdvDVP_InitSDK

"C" export
int AdvDVP_CloseSDK_C(DWORD dwObj)

3.2.4.2 Encode control

AdvDVP_InitEncoder

Syntax
int AdvDVP_InitEncoder(int nEncNum, int nEncBufSize, int VideoCompressFormat,
int AudioCompressFormat)

Parameters
[IN] nEncNum: Specifies the encoder number.
[IN] nEncBufSize: Specifies the encoding buffer size.
[IN] VideoCompressFormat: Specifies the format of video compression.
[IN] AudioCompressFormat: Specifies the format of audio compression.

Return Value
ENC_SUCCEEDED: Function succeeded.
ENC_FAILED: Function failed.
ENC_SDKINITFAILED: SDK is not initialized successfully.
ENC_ENCNUMERROR: Invalid encoder number.

Description

This function opens and initializes the specified video encoder. After initializing the encoder, the encoding state would be set as "ENC_STOPPED".

See Also

AdvDVP_CloseEncoder

AdvDVP_GetState

"C" export

int AdvDVP_InitEncoder_C(DWORD dwObj, int nEncNum, int nEncBufSize,
int VideoCompressFormat, int AudioCompressFormat)

AdvDVP_CloseEncoder

Syntax

int AdvDVP_CloseEncoder(int nEncNum)

Parameters

[IN] nEncNum: Specifies the encoder number.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.

Description

This function closes and releases the specified video encoder. After successfully calling this function, the encoding state would be set as "ENC_UNINITIALIZED".

See Also

AdvDVP_InitEncoder

AdvDVP_GetState

"C" export

int AdvDVP_CloseEncoder_C(DWORD dwObj, int nEncNum)

AdvDVP_StartVideoEncode

Syntax

int AdvDVP_StartVideoEncode(int nEncNum)

Parameters

[IN] nEncNum: Specifies the encoder number.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.

Description

This function notifies the specified video encoder to prepare to encode the video. The encode state would be set as "ENC_RUNNING" after a successful beginning.

See Also

AdvDVP_VideoEncode

AdvDVP_StopVideoEncode

AdvDVP_GetState

"C" export

int AdvDVP_StartVideoEncode_C(DWORD dwObj, int nEncNum)

AdvDVP_VideoEncode

Syntax

int AdvDVP_VideoEncode(int nEncNum, LPVOID lpInBuf, int InBufSize,
BOOL bKeyFrame)

Parameters

[IN] nEncNum: Specifies the encoder number.

[IN] lpInBuf: Specifies the input buffer that stores the source video frame.

[IN] InBufSize: Specifies the size of the input buffer.

[IN] bKeyFrame: Specifies if the video frame is encoded as a I-frame.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.
ENC_PARAMERROR:	Parameter error.

ENC_BUFFERFULL:	Encoding buffer is full, the video frame cannot be written to the buffer.

Description

This function writes the video frame to the encoding buffer to encode it by the specified encoder.

See Also

AdvDVP_StartVideoEncode

AdvDVP_StopVideoEncode

"C" export

int AdvDVP_VideoEncode_C(DWORD dwObj, int nEncNum, LPVOID lpInBuf, int InBufSize, BOOL bKeyFrame)

AdvDVP_StopVideoEncode

Syntax

int AdvDVP_StopVideoEncode(int nEncNum)

Parameters

[IN] nEncNum: Specifies the encoder number.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.

Description

This function notifies the specified video encoder to stop encoding and releases all relational resources. The encoding state would be set as "ENC_STOPPED" after a successful stop.

See Also

AdvDVP_StartVideoEncode

AdvDVP_VideoEncode

AdvDVP_GetState

"C" export

int AdvDVP_StopVideoEncode_C(DWORD dwObj, int nEncNum)

AdvDVP_GetState

Syntax

int AdvDVP_GetState(int nEncNum)

Parameters

[IN] nEncNum: Specifies the encoder number.

Return Value

ENC_SDKINITFAILED: SDK is not initialized successfully.
ENC_ENCNUMERROR: Invalid encoder number.

Description

This function gets encoding state of a specified video encoder.

```
typedef enum
{
        ENC_STOPPED         = 1,
        ENC_RUNNING         = 2,
        ENC_UNINITIALIZED   = -1,
} EncoderState;
```

See Also

AdvDVP_InitEncoder
AdvDVP_CloseEncoder
AdvDVP_StartVideoEncode
AdvDVP_StopVideoEncode

"C" export

int AdvDVP_GetState_C(DWORD dwObj, int nEncNum)

AdvDVP_SetStreamReadCB

Syntax

int AdvDVP_SetStreamReadCB(STREAMREAD_STRUCT *pStreamRead)

Parameters

[IN] pStreamRead: A pointer to STREAMREAD_STRUCT structure recording the pointers to the "StreamRead" callback functions.

Return Value

ENC_PARAMERROR: Parameter error.

Description

This function registers the "StreamRead" callback functions to the SDK.

See Also

STREAMREAD_STRUCT structure

"C" export

int AdvDVP_SetStreamReadCB_C(DWORD dwObj, STREAMREAD_STRUCT *pStreamRead)

AdvDVP_StartAudioEncode

Syntax

int AdvDVP_StartAudioEncode(int nEncNum)

Parameters

[IN] nEncNum: Specifies the encoder number.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.

Description

This function notifies the specified audio encoder to prepare to encode the video. The encode state would be set as "ENC_RUNNING" after a successful beginning.

See Also

AdvDVP_AudioEncode

AdvDVP_StopAudioEncode

"C" export

int AdvDVP_StartAudioEncode_C(DWORD dwObj, int nEncNum)

AdvDVP_AudioEncode

Syntax

int AdvDVP_AudioEncode(int nEncNum, LPVOID lpInBuf, int InBufSize)

Parameters

[IN] nEncNum: Specifies the encoder number.

[IN] lpInBuf: Specifies the input buffer that stores the source audio frame.

[IN] InBufSize: Specifies the size of the input buffer.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.
ENC_PARAMERROR:	Parameter error.
ENC_BUFFERFULL:	Encoding buffer is full, the video frame can not be written to the buffer.

Description

This function writes the audio frame to the encoding buffer to encode it by the specified encoder.

See Also

AdvDVP_StartAudioEncode

AdvDVP_StopAudioEncode

"C" export

int AdvDVP_AudioEncode_C(DWORD dwObj, int nEncNum, LPVOID lpInBuf, int InBufSize)

AdvDVP_StopAudioEncode

Syntax

int AdvDVP_StopAudioEncode(int nEncNum)

Parameters

[IN] nEncNum: Specifies the encoder number.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.

Description

This function notifies the specified audio encoder to stop encoding and releases all relational resources. The encoding state would be set as "ENC_STOPPED" after a successful stop.

See Also

AdvDVP_StartAudioEncode

AdvDVP_AudioEncode

"C" export

int AdvDVP_StopAudioEncode_C(DWORD dwObj, int nEncNum)

3.2.4.3 Encode setting

AdvDVP_GetVideoQuant

Syntax

int AdvDVP_GetVideoQuant(int nEncNum, int *nQuant)

Parameters

[IN] nEncNum: Specifies the encoder number.

[OUT] nQuant: A pointer to get the video quant. The range is 1~31.
The default video quality is 4.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.
ENC_PARAMERROR:	The nQuant parameter is invalid.

Description

This function gets video quant of the specified video encoder. The lower video quant can get the compressed video with higher quality and bit rate, vice versa.

See Also

AdvDVP_SetVideoQuant

"C" export

int AdvDVP_GetVideoQuant_C(DWORD dwObj, int nEncNum, int *nQuant)

AdvDVP_SetVideoQuant

Syntax

int AdvDVP_SetVideoQuant(int nEncNum, int nQuant)

Parameters

[IN] nEncNum: Specifies the encoder number.

[IN] nQuant: A value to set the video quant. The range is 1~31.
The default video quality is 4.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.

Description

This function sets video quant of the specified video encoder. The lower video quant can get the compressed video with higher quality and bit rate, vice versa.

See Also

AdvDVP_GetVideoQuant

"C" export

int AdvDVP_SetVideoQuant_C(DWORD dwObj, int nEncNum, int nQuant)

AdvDVP_GetVideoFrameRate

Syntax

int AdvDVP_GetVideoFrameRate(int nEncNum, int *nFrameRate)

Parameters

[IN] nEncNum: Specifies the encoder number.

[OUT] nFrameRate: A pointer to get the video frame rate.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.
ENC_PARAMERROR:	The nFrameRate parameter is invalid.

Description

This function gets video frame rate of the specified video encoder.

See Also

AdvDVP_SetVideoFrameRate

"C" export

int AdvDVP_GetVideoFrameRate_C(DWORD dwObj, int nEncNum,
int *nFrameRate)

AdvDVP_SetVideoFrameRate

Syntax

int AdvDVP_SetVideoFrameRate(int nEncNum, int nFrameRate)

Parameters

[IN] nEncNum: Specifies the encoder number.

[IN] nFrameRate: A value to set the video frame rate. The range is 1~30/NTSC (1~25/PAL). The default video frame rate is 30/NTSC(25/PAL).

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.

Description

This function sets video frame rate of the specified video encoder. The set value of this function will come into effect only before AdvDVP_StartVideoEncoder is called. Calling this function after AdvDVP_StartVideoEncoder will not take effect, but you can use AdvDVP_GetVideoFrameRate to get the latest value you have set.

See Also

AdvDVP_GetVideoFrameRate

"C" export

int AdvDVP_SetVideoFrameRate_C(DWORD dwObj, int nEncNum, int nFrameRate)

AdvDVP_GetVideoResolution

Syntax

int AdvDVP_GetVideoResolution(int nEncNum, int *nWidth, int *nHeight)

Parameters

[IN] nEncNum: Specifies the encoder number.
[OUT] nWidth: A pointer to get the width of the video.
[OUT] nHeight: A pointer to get the height of the video.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.
ENC_PARAMERROR:	Invalid pointer parameter.

Description

This function gets video resolution of the specified video encoder.

See Also

AdvDVP_SetVideoResolution

"C" export

int AdvDVP_GetVideoResolution_C(DWORD dwObj, int nEncNum, int *nWidth, int *Height)

AdvDVP_SetVideoResolution

Syntax

int AdvDVP_SetVideoResolution(int nEncNum, int nWidth, int nHeight)

Parameters

[IN] nEncNum: Specifies the encoder number.

[IN] nWidth: A value to set the width of video. The default width is 320.

[IN] nHeight A value to set the height of video. The default height is 240.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.

Description

This function sets video resolution of the specified video encoder. The set value of this function will come into effect only before AdvDVP_StartVideoEncoder is called. Calling this function after AdvDVP_StartVideoEncoder will not take effect, but you can use AdvDVP_GetVideoResolution to get the latest value you have set.

See Also

AdvDVP_GetVideoResolution

"C" export

int AdvDVP_SetVideoResolution_C(DWORD dwObj, int nEncNum, int nWidth,
int nHeight)

AdvDVP_GetVideoKeyInterval

Syntax

int AdvDVP_GetVideoKeyInterval(int nEncNum, int *nKeyInterval)

Parameters

[IN] nEncNum: Specifies the encoder number.

[OUT] nKeyInterval: A pointer to get the interval of the video key frame.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.
ENC_PARAMERROR:	The nKeyInterval parameter is invalid.

Description

This function gets the interval of the video key frame of the specified video encoder.

See Also

AdvDVP_SetVideoKeyInterval

"C" export

int AdvDVP_GetVideoKeyInerval_C(DWORD dwObj, int nEncNum,
int *nKeyInterval)

AdvDVP_SetVideoKeyInterval

Syntax

int AdvDVP_SetVideoKeyInterval(int nEncNum, int nKeyInterval)

Parameters

[IN] nEncNum: Specifies the encoder number.

[IN] nKeyInterval: A value to set the interval of the video key frame. The range is 1~300. The default interval of video key frame is 100. If the value is less than 1, it will use 1. If the value is greater than 300, it will use 300.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.

Description

This function sets the interval of the video key frame of the specified video encoder.

See Also

AdvDVP_GetVideoKeyInterval

"C" export

int AdvDVP_SetVideoKeyInterval_C(DWORD dwObj, int nEncNum, int nKeyInterval)

AdvDVP_GetVideoQuality

Syntax

int AdvDVP_GetVideoQuality(int nEncNum, int *nQuality)

Parameters

[IN] nEncNum: Specifies the encoder number.

[OUT] nQuality: A pointer to get the video quality. The range is 0~99.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.
ENC_PARAMERROR:	The nQuant parameter is invalid.

Description

This function gets video quality of the specified video encoder. The higher video quality can get the compressed video with higher quality and bit rate, vice versa.

See Also

AdvDVP_SetVideoQuality

"C" export

int AdvDVP_GetVideoQuality_C(DWORD dwObj, int nEncNum, int *nQuality)

AdvDVP_SetVideoQuality

Syntax

int AdvDVP_SetVideoQuality(int nEncNum, int nQuality)

Parameters

[IN] nEncNum: Specifies the encoder number.

[IN] nQuality: A value to set the video quant. The range is 0~99.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.

Description

This function sets video quant of the specified video encoder. The higher video quality can get the compressed video with higher quality and bit rate, vice versa.

See Also

AdvDVP_GetVideoQuality

"C" export

int AdvDVP_SetVideoQuality_C(DWORD dwObj, int nEncNum, int nQuality)

AdvDVP_GetVideoCompressFormats

Syntax

int AdvDVP_GetVideoCompressFormats(COMPRESS_FORMAT_STRUCT *pCompressFormats, int nMaxFormats)

Parameters

[IN] pCompressFormats: A pointer to the COMPRESS_FORMAT_STRUCT structure array buffer.

[IN] nMaxFormats: The number of COMPRESS_FORMAT_STRUCT structure array buffer.

Return Value

Returns the number for compression format, or the following error codes,

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.

Description

This function gets all of the available video compression formats of software encoding.

See Also

AdvDVP_GetAudioCompressFormats

"C" export

```
int AdvDVP_GetVideoCompressFormats_C(DWORD dwObj,
                                     COMPRESS_FORMAT_STRUCT *pCompressFormats,
                                     int nMaxFormats)
```

AdvDVP_GetCurVideoCompressFormat

Syntax

```
int AdvDVP_GetCurVideoCompressFormat(int nEncNum)
```

Parameters

[IN] nEncNum: Specifies the encoder number.

Return Value

Returns the 'VideoCompressFormat' constant of compression format, or the following error codes,

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.

Description

This function gets the video compression format of software encoding.

See Also

AdvDVP_GetVideoCompressFormats

AdvDVP_GetCurAudioCompressFormat

"C" export

```
int AdvDVP_GetCurVideoCompressFormat_C(DWORD dwObj, int nEncNum)
```

AdvDVP_GetAudioCompressFormats

Syntax

int AdvDVP_GetAudioCompressFormats(COMPRESS_FORMAT_STRUCT *pCompressFormats, int nMaxFormats)

Parameters

[IN] pCompressFormats: A pointer to the COMPRESS_FORMAT_STRUCT structure array buffer.

[IN] nMaxFormats: The number of COMPRESS_FORMAT_STRUCT structure array buffer.

Return Value

Returns the number for compression format, or the following error codes,

ENC_SUCCEEDED: Function succeeded.

ENC_FAILED: Function failed.

ENC_SDKINITFAILED: SDK is not initialized successfully.

ENC_ENCINITFAILED: Encoder is not initialized successfully.

Description

This function gets all of the available audio compression formats of software encoding.

See Also

AdvDVP_GetVideoCompressFormats

"C" export

int AdvDVP_GetAudioCompressFormats_C(DWORD dwObj, COMPRESS_FORMAT_STRUCT *pCompressFormats, int nMaxFormats)

AdvDVP_GetCurAudioCompressFormat

Syntax

int AdvDVP_GetCurAudioCompressFormat(int nEncNum)

Parameters

[IN] nEncNum: Specifies the encoder number.

Return Value

Returns the 'AudioCompressFormat' constant of compression format, or the following error codes,

ENC_SUCCEEDED: Function succeeded.

ENC_FAILED: Function failed.

ENC_SDKINITFAILED: SDK is not initialized successfully.

ENC_ENCINITFAILED: Encoder is not initialized successfully.

Description

This function gets the audio compression format of software encoding.

See Also

AdvDVP_GetAudioCompressFormats

AdvDVP_GetCurVideoCompressFormat

"C" export

int AdvDVP_GetCurAudioCompressFormat_C(DWORD dwObj, int nEncNum)

AdvDVP_GetAudioFormat

Syntax

int AdvDVP_GetAudioFormat(int nDevNum, int *pnChannel, int *pnBitsPerSample, int *pnSamplePerSec)

Parameters

[IN] nEncNum: Specifies the encoder number.

[OUT] pnChannel: A pointer to get the channel number of audio.

[OUT] pnBitsPerSample: A pointer to get the bits number of audio sample.

[OUT] pnSamplePerSec: A pointer to get the audio sampling number for each second.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.
ENC_PARAMERROR:	Invalid pointer parameter.

Description

This function gets audio input format of a specified capture port.

See Also

AdvDVP_SetAudioFormat

"C" export

int AdvDVP_GetAudioFormat_C(DWORD dwObj, int nDevNum, int *pnChannel, int *pnBitsPerSample, int *pnSamplePerSec)

AdvDVP_SetAudioFormat

Syntax

int AdvDVP_SetAudioFormat(int nDevNum, int nChannel, int nBitsPerSample,
int nSamplePerSec)

Parameters

[IN] nEncNum: Specifies the encoder number.

[IN] nChannel: The channel number of audio. 1 is mono, 2 is stereo.

[IN] nBitsPerSample: The bits number of audio sample. Default is 16.

[IN] nSamplePerSec: The audio sampling number for each second. Default is 8000.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_ENCNUMERROR:	Invalid encoder number.
ENC_ENCINITFAILED:	Encoder is not initialized successfully.

Description

This function sets audio input format of a specified capture port.

See Also

AdvDVP_GetAudioFormat

"C" export

int AdvDVP_SetAudioFormat_C(DWORD dwObj, int DevNum, int nChannel,
int nBitsPerSample, int nSamplePerSec)

3.2.4.4 De-interlace control

AdvDVP_SetDeinterlace

Syntax

int AdvDVP_SetDeinterlace(BOOL bEnable)

Parameters

[IN] bEnable: TRUE for enable; False for disable.

Return Value

Return 1 means 'De-Interlace' is enabled, otherwise returns 0.

Description

This function enables or disables the de-interlace function.

See Also

AdvDVP_GetDeinterlace

"C" export

int AdvDVP_SetDeinterlace_C(DWORD dwObj, BOOL bEnable)

AdvDVP_GetDeinterlace

Syntax

int AdvDVP_GetDeinterlace()

Parameters

None.

Return Value

Return 1 means 'De-Interlace' is enabled, otherwise returns 0.

Description

This function retrieves de-interlace state.

See Also

AdvDVP_SetDeinterlace

"C" export

int AdvDVP_GetDeinterlace_C(DWORD dwObj)

3.2.4.5 AVI control

AdvDVP_CreateAVIFile

Syntax

HANDLE AdvDVP_CreateAVIFile(LPCSTR lpcsFileName, int nWidth, int nHeight,
int nFrameRate, int CompressFormat)

Parameters

[IN] lpcsFileName: Specifies the file name of the AVI file.

[IN] nWidth: The width of the image.

[IN] nHeight: The height of the image.

[IN] nFrameRate: Specifies the frame rate of the video.

[IN] CompressFormat: Specifies the format of video compression.

```
typedef enum
{
        VID_COMP_NONE               = -1,
        VID_COMP_RMP4               = 0,    // S/W compression
        VID_COMP_MPEG4_DIVX         = 1,
        VID_COMP_MPEG4_XVID         = 2,
        VID_COMP_X264               = 3,
        VID_COMP_MPG4_SW            = 4,
        VID_COMP_H264_SW            = 5,
        VID_COMP_HW_COMPRESS        = 10,
        VID_COMP_JPEG               = 10,   // H/W compression
        VID_COMP_MPEG2              = 11,
        VID_COMP_MPEG4              = 12,
        VID_COMP_H264_BaseLine      = 13,
        VID_COMP_H264_MainProfile   = 14,   // CABAC
        VID_COMP_H264_MainProfile2  = 15,   // CALVC
        VID_COMP_H264_HighProfile   = 16,   // CABAC
        VID_COMP_H264_HighProfile2  = 17,   // CALVC
} VideoCompressFormat;
```

Return Value

If the function succeeds, the file handle is returned. Otherwise, the function returns NULL.

Description

This function creates the AVI file to save the encoded video stream.

See Also

AdvDVP_WriteAVIFile

AdvDVP_CloseAVIFile

AdvDVP_AudioToAVIFile

"C" export

HANDLE AdvDVP_CreateAVIFile_C(DWORD dwObj, LPCSTR lpcsFileName,
int nWidth, int nHeight, int nFrameRate,
int CompressFormat)

AdvDVP_WriteAVIFile

Syntax

int AdvDVP_WriteAVIFile(HANDLE hAVIFile, LPVOID lpStreamBuf, long lBufSize, DWORD dwCompFlags)

Parameters

[IN] hAVIFile: Specifies the AVI file handle.

[IN] lpStreamBuf: A pointer to the video stream data buffer written into the file.

[IN] lBufSize: Specifies the size of the video stream data buffer.

[IN] dwCompFlags: Flag associated with this data. The AVIIF_KEYFRAME flag is defined to indicate that this data does not rely on preceding data in the file.

#define AVIIF_KEYFRAME 0x00000010L

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.
ENC_PARAMERROR:	The lpStreamBuf is NULL.

Description

This function writes the video stream data into the specified AVI file.

See Also

AdvDVP_CreateAVIFile

AdvDVP_CloseAVIFile

"C" export

int AdvDVP_WriteAVIFile_C(DWORD dwObj, HANDLE hAVIFile, LPVOID lpStreamBuf, long lBufSize, DWORD dwCompFlags)

AdvDVP_CloseAVIFile

Syntax

int AdvDVP_CloseAVIFile(HANDLE hAVIFile)

Parameters

[IN] hAVIFile: Specifies the AVI file handle.

Return Value

ENC_SUCCEEDED:	Function succeeded.
ENC_FAILED:	Function failed.
ENC_SDKINITFAILED:	SDK is not initialized successfully.

Description

This function closes the specified AVI file.

See Also

AdvDVP_CreateAVIFile
AdvDVP_WriteAVIFile

"C" export

int AdvDVP_CloseAVIFile_C(DWORD dwObj, HANDLE hAVIFile)

AdvDVP_AudioToAVIFile

Syntax

HANDLE AdvDVP_AudioToAVIFile(HANDLE hAVIFile, int nSampleRate, int nChannel, int nBitPerSample, int CompressFormat)

Parameters

[IN] hAVIFile: Specifies the AVI file handle.
[IN] nSampleRate: The sampling rate of the audio.
[IN] nChannel: The channel number of of the audio. 1: mono, 2: stereo.
[IN] nBitPerSample: The bits number of each audio sample.
[IN] CompressFormat: Specifies the format of audio compression.

```
typedef enum
{
        AUD_COMP_PCM      = 0,
        AUD_COMP_ALAW     = 1,
        AUD_COMP_MULAW    = 2,
        AUD_COMP_AC3      = 3,
} AudioCompressFormat;
```

Return Value

If the function succeeds, the file handle is returned. Otherwise, the function returns NULL.

Description

This function creates the audio track in AVI file to save the encoded audio stream.

See Also

AdvDVP_CreateAVIFile
AdvDVP_CloseAVIFile

"C" export

```
int AdvDVP_AudioToAVIFile_C(DWORD dwObj, HANDLE hAVIFile,
                            int nSampleRate, int nChannel, int nBitPerSample,
                             int CompressFormat)
```

3.3 Playback APIs

AdvDVP- Series Functions Reference

Library: DVP7010BPlay.DLL

3.3.1 Data Type

PlayerRes

Syntax

```
typedef enum tagRes
{
PLAYER_SUCCEEDED        = 1,
PLAYER_FAILED           = 0,
PLAYER_SDKINITFAILED    = -1,
PLAYER_PARAMERROR       = -2,
} PlayerRes;
```

Description

The function returns value.

EPlayerStatus

Syntax

```
typedef enum tagRes
{
PLAYER_NOTOPENED   = 0,
PLAYER_OPENED      = 1,
PLAYER_PLAYING     = 2,
PLAYER_STOPPED     = 3,
PLAYER_PAUSED      = 4,
} PlayerRes;
```

Description

The function returns value.

3.3.2 Methods

3.3.2.1 Initialize and close

AdvDVP_CreatePlayerSDKInstance

Syntax

int AdvDVP_CreatePlayerSDKInstance(void **pp)

Parameters

[OUT] pp: A pointer to the player SDK instance.

Return Value

PLAYER_SUCCEEDED:	Function succeeded.
PLAYER_FAILED:	Function failed.
PLAYER_PARAMERROR:	Parameter error.

Description

This function creates "playback" SDK instance. You can delete the pointer which is carried by the parameter named "pp" to free this SDK instance before you free the DLL handle.

See Also

"C" export

DWORD AdvDVP_CreatePlayerSDKInstance()

return a DWORD value to indicate the instance of player SDK. return NULL meaning function failed.

AdvDVP_OpenFile

Syntax

int AdvDVP_OpenFile(LPCSTR lpcsFileName)

Parameters

[IN] lpcsFileName: Specifies the file name of the source video file.

Return Value

PLAYER_SUCCEEDED: Function succeeded.

PLAYER_FAILED: Function failed.

Description

This function opens the source video file and initializes the video player. The playback status would be set as "PLAYER_STOPPED" after successfully calling this function.

See Also

AdvDVP_CloseFile

AdvDVP_GetStatus

"C" export

int AdvDVP_OpenFile_C(DWORD dwObj, LPCSTR lpcsFileName)

AdvDVP_CloseFile

Syntax

int AdvDVP_CloseFile()

Parameters

None.

Return Value

PLAYER_SUCCEEDED: Function succeeded.

PLAYER_FAILED: Function failed.

Description

This function closes the source video file and free resources allocated for video player. The playback status would be set as "PLAYER_NOTOPENED" after successfully calling this function.

See Also

AdvDVP_OpenFile

AdvDVP_GetStatus

"C" export

int AdvDVP_CloseFile_C(DWORD dwObj)

3.3.2.2 Playback control

AdvDVP_Play

Syntax

int AdvDVP_Play(HWND hwndApp, BOOL bAutoResizeWnd)

Parameters

[IN] hwndApp: A windows handle for display area.

[IN] bAutoResizeWnd: Specifies if the display area is resized automatically according to the video resolution.

Return Value

PLAYER_SUCCEEDED: Function succeeded.

PLAYER_FAILED: Function failed.

Description

This function plays the file that has been opened. The playback status would be set as "PLAYER_PLAYING" after successfully calling this function.

See Also

AdvDVP_Pause

AdvDVP_Stop

AdvDVP_GetStatus

"C" export

int AdvDVP_Play_C(DWORD dwObj, HWND hwndApp, BOOL bAutoResizeWnd)

AdvDVP_Pause

Syntax

int AdvDVP_Pause()

Parameters

None.

Return Value

PLAYER_SUCCEEDED: Function succeeded.

PLAYER_FAILED: Function failed.

Description

This function pauses or continues the file that has been opened. The playback status would be set as "PLAYER_PAUSED" after successfully calling this function.

See Also

AdvDVP_Play

AdvDVP_Stop

AdvDVP_GetStatus

"C" export

int AdvDVP_Pause_C(DWORD dwObj)

AdvDVP_Stop

Syntax

int AdvDVP_Stop()

Parameters

None.

Return Value

PLAYER_SUCCEEDED: Function succeeded.

PLAYER_FAILED: Function failed.

Description

This function stops the file that is playing. The playback status would be set as "PLAYER_STOPPED" after successfully calling this function.

See Also

AdvDVP_Play

AdvDVP_Pause

AdvDVP_GetStatus

"C" export

int AdvDVP_Stop_C(DWORD dwObj)

AdvDVP_Fast

Syntax

int AdvDVP_Fast()

Parameters

None.

Return Value

PLAYER_SUCCEEDED: Function succeeded.

PLAYER_FAILED: Function failed.

Description

This function improves the current play speed by one time, 4 times at most. The playback status would be set as "PLAYER_PLAYING" after successfully calling this function.

See Also

AdvDVP_Pause

AdvDVP_Slow

AdvDVP_GetStatus

"C" export

int AdvDVP_Fast_C(DWORD dwObj)

AdvDVP_Slow

Syntax

int AdvDVP_Slow()

Parameters

None.

Return Value

PLAYER_SUCCEEDED: Function succeeded.

PLAYER_FAILED: Function failed.

Description

This function slows the current play speed by one time, 4 times at most. The playback status would be set as "PLAYER_PLAYING" after successfully calling this function.

See Also

AdvDVP_Pause

AdvDVP_Fast

AdvDVP_GetStatus

"C" export

int AdvDVP_Slow_C(DWORD dwObj)

AdvDVP_PlayStep

Syntax

int AdvDVP_PlayStep()

Parameters

None.

Return Value

PLAYER_SUCCEEDED:	Function succeeded.
PLAYER_FAILED:	Function failed.

Description

This function makes the video to step forward one frame. The playback status would be set as "PLAYER_PAUSED" after successfully calling this function. This function should sleep for a little time, for example, 100 milliseconds before calling AdvDVP_GetStatus.

See Also

AdvDVP_Fast

AdvDVP_Slow

AdvDVP_GetStatus

"C" export

int AdvDVP_PlayStep_C(DWORD dwObj)

AdvDVP_GetStatus

Syntax

int AdvDVP_GetStatus()

Parameters

None.

Return Value

PLAYER_SUCCEEDED: Function succeeded.
PLAYER_FAILED: Function failed.

Description

This function gets playback status.

```
typedef enum tagPlayerStatus{
    PLAYER_NOTOPENED    = 0,
    PLAYER_OPENED       = 1,
    PLAYER_PLAYING      = 2,
    PLAYER_STOPPED      = 3,
    PLAYER_PAUSED       = 4
} PlayerStatus;
```

See Also

AdvDVP_OpenFile
AdvDVP_CloseFile
AdvDVP_Play
AdvDVP_Pause
AdvDVP_Stop
AdvDVP_Fast
AdvDVP_Slow
AdvDVP_PlayStep

"C" export

int AdvDVP_GetStatus_C(DWORD dwObj)

AdvDVP_GetCurImage

Syntax

int AdvDVP_GetCurImage(LPBYTE *lpImage, long *pBufSize)

Parameters

[OUT] lpImage: A pointer to a image buffer.

[IN/OUT] pBufSize: A long pointer to receive the returned image buffer size.

Return Value

PLAYER_SUCCEEDED:	Function succeeded.
PLAYER_FAILED:	Function failed.
PLAYER_PARAMERROR:	Parameter error.

Description

This function gets current played video image.

See Also

"C" export

int AdvDVP_GetCurImage_C(DWORD dwObj, LPBYTE *lpImage, long *pBufSize)

AdvDVP_RegNotifyMsg

Syntax

int AdvDVP_RegNotifyMsg(HWND hWnd, UINT nMsg)

Parameters

[IN] hWnd: Specifies the handle of the window receiving this message.

[IN] nMsg: Specifies the user-defined message.

When this message is received, it means some event of the playback occurs, such as the file playing is finished.

Return Value

PLAYER_SUCCEEDED: Function succeeded.

PLAYER_FAILED: Function failed.

Description

This function registers a user-defined message. When an event of the playback occurs, this message will be sent to the specified window. This function must be called after "AdvDVP_OpenFile" function.

See Also

AdvDVP_CheckFileEnd

"C" export

int AdvDVP_RegNotifyMsg_C(DWORD dwObj, HWND hwnd, UINT nMsg)

AdvDVP_CheckFileEnd

Syntax

BOOL AdvDVP_CheckFileEnd()

Parameters

None.

Return Value

If the event that the file playing end is detected, this function returns TRUE. Otherwise, the function returns FALSE.

Description

This function checks if the file playing is finished.

See Also

AdvDVP_RegNotifyMsg

"C" export

BOOL AdvDVP_CheckFileEnd_C(DWORD dwObj)

3.3.2.3 Playback setting

AdvDVP_GetVideoResolution

Syntax

int AdvDVP_GetVideoResolution(int *nWidth, int *nHeight)

Parameters

[OUT] nWidth: An integer pointer to get the width of the video.

[OUT] nHeight: An integer pointer to get the height of the video.

Return Value

PLAYER_SUCCEEDED: Function succeeded.

PLAYER_FAILED: Function failed.

Description

This function gets the width and height of the video.

See Also

"C" export

int AdvDVP_GetVideoResolution_C(DWORD dwObj, int *nWidth, int *nHeight)

AdvDVP_GetFileTime

Syntax

double AdvDVP_GetFileTime()

Parameters

None.

Return Value

If the function succeeded, the total file time is returned. Otherwise, the function returns 0.

Description

This function retrieves total file time in seconds.

See Also

AdvDVP_GetPlayedTime

AdvDVP_SetPlayPosition

"C" export

double AdvDVP_GetFileTime_C(DWORD dwObj)

AdvDVP_GetPlayedTime

Syntax

double AdvDVP_GetPlayedTime()

Parameters

None.

Return Value

If the function succeeded, the current file time is returned. Otherwise, the function returns 0.

Description

This function retrieves current file time in seconds.

See Also

AdvDVP_GetFileTime

AdvDVP_SetPlayPosition

"C" export

double AdvDVP_GetPlayedTime_C(DWORD dwObj)

AdvDVP_SetPlayPosition

Syntax

int AdvDVP_SetPlayPosition(double dTime)

Parameters

[IN] dTime: Specifies the file time in seconds.

Return Value

PLAYER_SUCCEEDED: Function succeeded.

PLAYER_FAILED: Function failed.

Description

This function adjusts the file time to the specified file time. If you want to call AdvDVP_CheckFileEnd after this function, this function should sleep at least 500 milliseconds before AdvDVP_CheckFileEnd.

See Also

AdvDVP_GetFileTime

AdvDVP_GetPlayedTime

"C" export

int AdvDVP_SetPlayPosition_C(DWORD dwObj, double dTime)

AdvDVP_GetFileTotalFrames

Syntax

LONGLONG AdvDVP_GetFileTotalFrames()

Parameters

None.

Return Value

If the function succeeded, the total number of the frames in the file is returned. Otherwise, the function returns 0.

Description

This function retrieves total number of the frames in the file.

See Also

AdvDVP_GetPlayedFrames

"C" export

LONGLONG AdvDVP_GetFileTotalFrames_C(DWORD dwObj)

AdvDVP_GetPlayedFrames

Syntax

LONGLONG AdvDVP_GetPlayedFrames()

Parameters

None

Return Value

If the function succeeded, the current frame number of the file is returned. Otherwise, the function returns 0.

Description

This function retrieves current frame number of the file.

See Also

AdvDVP_GetFileTotalFrames

"C" export

LONGLONG AdvDVP_GetPlayedFrames_C(DWORD dwObj)

AdvDVP_GetPlayRate

Syntax

double AdvDVP_GetPlayRate()

Parameters

None.

Return Value

If the function succeeded, the playback ratio is returned. Otherwise, the function returns 0.

Description

This function retrieves the playback rate.

See Also

AdvDVP_GetFileTime

AdvDVP_GetPlayedTime

"C" export

double AdvDVP_GetPlayRate_C(DWORD dwObj)

ADVANTECH

Enabling an Intelligent Planet

www.advantech.com

Please verify specifications before quoting. This guide is intended for reference purposes only.

All product specifications are subject to change without notice.

No part of this publication may be reproduced in any form or by any means, electronic, photocopying, recording or otherwise, without prior written permission of the publisher.

All brand and product names are trademarks or registered trademarks of their respective companies.

© Advantech Co., Ltd. 2012